

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

INTEGRACIÓN DE UN SISTEMA DE RECONOCIMIENTO DE GESTOS DE LA MANO BASADO EN EL SENSOR MYO ARMBAND CON EL SIMULADOR DE ROBOTS COPPELIASIM PARA EL CONTROL DE UN MANIPULADOR VIRTUAL DE 6 GRADOS DE LIBERTAD

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN ELECTRÓNICA Y CONTROL**

CHICO GODOY ALEX PAÚL

alex.chico@epn.edu.ec

DIRECTOR: ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

patricio.cruz@epn.edu.ec

Quito, Marzo 2022

AVAL

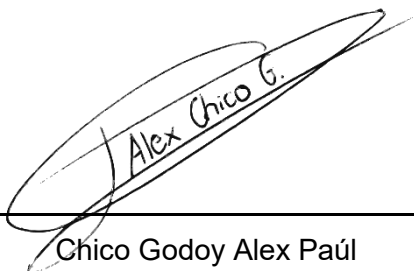
Certifico que el presente trabajo fue desarrollado por Alex Paúl Chico Godoy, bajo mi supervisión.

ING. PATRICIO JAVIER CRUZ DÁVALOS, PhD.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Chico Godoy Alex Paúl, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.



Alex Chico G.

Chico Godoy Alex Paúl

DEDICATORIA

A mis padres Luis Raúl Chico y Bety Godoy.

Alex

AGRADECIMIENTO

Agradezco a mis padres y hermanos que siempre me brindaron su apoyo y confianza incondicional durante el transcurso de mi vida, especialmente en mi etapa universitaria, y que espero nunca decepcionarlos.

A mis abuelitos por todo su esfuerzo y apoyo que hicieron posible alcanzar este objetivo.

Al Dr. Patricio J. Cruz por su guía y apoyo fundamental que hicieron posible el desarrollo del presente proyecto.

Al Dr. Juan Pablo Vásquez y de igual manera al Ing. Robert Guamán por guiarme y brindarme su ayuda en el desarrollo de este proyecto.

A todos los profesores que fueron parte fundamental de mi formación académica, por brindarme no solo conocimientos, sino también consejos y mensajes de superación.

A todos los amigos que me acompañaron en mi vida universitaria, con quienes he compartido los momentos más felices y difíciles de mi vida.

A todas las demás personas que me acompañaron durante esta etapa, y siempre me brindaron apoyo y mensajes de aliento para siempre seguir adelante.

Alex

ÍNDICE DE CONTENIDO

AVAL	I
DECLARACIÓN DE AUTORÍA.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	IV
ÍNDICE DE CONTENIDO.....	V
RESUMEN	VII
ABSTRACT	VIII
1 INTRODUCCIÓN.....	1
1.1 OBJETIVOS	2
1.2 ALCANCE	2
1.3 MARCO TEÓRICO.....	4
1.3.1 MANIPULADORES ROBÓTICOS	4
1.3.1.1 Transformaciones homogéneas.....	5
1.3.1.2 Cinemática de manipuladores.....	7
1.3.2 MANIPULADOR ROBÓTICO UR5	15
1.3.3 CONTROL DE MANIPULADORES ROBÓTICOS	16
1.3.3.1 Control PID de Mínima Norma	17
1.3.3.2 Controlador basado en Álgebra Lineal	19
1.3.3.3 Rendimiento de los Sistemas de Control.....	21
1.3.4 ENTORNO DE SIMULACIÓN COPPELIASIM EDU (V-REP)	22
1.3.4.1 API Coppeliasim	23
1.3.5 SISTEMAS DE RECONOCIMIENTO DE GESTOS	23
1.3.5.1 Señales Electromiográficas	24
1.3.5.2 Sensor Myo Armband	25
1.3.5.3 Sistema Desarrollado en el Proyecto de Investigación PIGR-19-07	26
2 METODOLOGÍA.....	33
2.1 ARQUITECTURA DEL SISTEMA	33
2.2 MODELO CINEMÁTICO DEL BRAZO ROBÓTICO UR5.....	34
2.2.1 MODELO CINEMÁTICO DIRECTO	34
2.2.2 MODELO CINEMÁTICO INVERSO.....	36
2.2.3 MATRIZ JACOBIANA.....	43
2.2.3.1 Matriz Jacobiana Analítica	44

2.3	DISEÑO DEL ENTORNO VIRTUAL EN COPPELIASIM	46
2.4	IMPLEMENTACIÓN DE LOS ALGORITMOS DE CONTROL.....	48
2.5	INTEGRACIÓN DEL SISTEMA DE RECONOCIMIENTO.....	51
2.5.1	GESTOS Y COMANDOS DE SELECCIÓN	52
2.5.2	COMANDOS DE MOVIMIENTO.....	53
2.6	DESARROLLO DE LA INTERFAZ GRÁFICA.....	54
	Control de Posición y Orientación.....	56
	Seguimiento de Trayectorias	57
	Modo HGR+IMU	57
3	RESULTADOS Y DISCUSIÓN	58
3.1	VALIDACIÓN DEL MODELO CINEMÁTICO.....	58
3.2	CALIBRACIÓN DE LOS CONTROLADORES	62
3.3	PRUEBAS SIN EL SISTEMA DE RECONOCIMIENTO	67
3.3.1	CONTROL DE POSICIÓN Y ORIENTACIÓN	67
3.3.1.1	Rendimiento de los Controladores	71
3.3.2	SEGUIMIENTO DE TRAYECTORIAS	72
3.3.2.1	Trayectoria Cuadrada	73
3.3.2.2	Trayectoria Circular.....	77
3.3.2.3	Rendimiento de los Controladores	79
3.4	PRUEBAS CON EL SISTEMA DE RECONOCIMIENTO	80
3.4.1	MODO HGR+IMU.....	80
3.4.1.1	Rendimiento de los Controladores	84
4	CONCLUSIONES Y RECOMENDACIONES.....	85
4.1	CONCLUSIONES.....	85
4.2	RECOMENDACIONES	86
5	REFERENCIAS BIBLIOGRÁFICAS	88
	ANEXOS	92

RESUMEN

En este trabajo se presenta el diseño, simulación y comparación de desempeño de dos técnicas de control, un controlador PID basado en mínima norma y un controlador basado en álgebra lineal, para un manipulador robótico UR5 virtual. Estos son integrados con el sistema de reconocimiento de gestos de la mano desarrollado en el Proyecto de Investigación PIGR-19-07 de la Escuela Politécnica Nacional, capaz de reconocer cinco gestos diferentes de la mano utilizando las señales electromiográficas adquiridas por el sensor Myo Armband, que se coloca en el antebrazo del usuario. El entorno virtual en donde se efectúan las pruebas de los controladores del robot manipulador se desarrolla en el software de simulación robótica CoppeliaSim.

Los controladores están diseñados en base al modelo cinemático del robot, por lo que también se describe el desarrollo de dicho modelo partiendo de sus relaciones geométricas básicas. Además, se analiza y se compara el rendimiento de los controladores a través del índice de la Integral del Error Absoluto (IAE) a partir de pruebas de funcionamiento con y sin el sistema de reconocimiento de gestos. Adicionalmente, se presenta la implementación de una interfaz gráfica desarrollada en App Designer de Matlab, la cual permite al usuario interactuar con la simulación del robot, elegir entre sus diferentes modos de funcionamiento, y monitorear su estado.

PALABRAS CLAVE: Manipulador Robótico UR5, CoppeliaSim, Sistema de Reconocimiento de Gestos, Myo Armband, Controlador PID, Controlador Algebraico.

ABSTRACT

This work presents the design, simulation and performance comparison of two control techniques: a PID controller based on minimum norm and a controller based on linear algebra, both for a virtual UR5 robotic manipulator. These controllers are integrated with the hand gesture recognition system developed in the PIGR-19-07 Research Project of the Escuela Politécnica Nacional. This system is capable of recognizing five different hand gestures using electromyographic signals acquired by a Myo Armband sensor, which is placed on the user's forearm. The virtual environment, where the simulation tests of the robot manipulator controllers are carried, is developed in the robotic simulation software CoppeliaSim.

The controllers are designed based on the kinematic model of the robot, so the development of this model is presented starting from its basic geometric relationships. The performance of each controller is also compared through the Integral Absolute Error (IAE) criterion from functional tests with and without the gesture recognition system. In addition, the implementation of a graphical interface developed in Matlab's App Designer is detailed, which allows the user to interact with the robot simulation, to choose between its different operating modes, and to monitor its status.

KEYWORDS: UR5 Robotic Manipulator, CoppeliaSim, Had-gesture Recognition System, Myo Armband, PID Controller, Linear Algebra based Controller.

1 INTRODUCCIÓN

En las últimas décadas, el uso de manipuladores robóticos se ha ido extendiendo en muchas aplicaciones como la intervención en entornos peligrosos, cirugía robótica, prótesis, montaje industrial, aviación, entre otras [1]. En estos escenarios, el operador humano es a menudo esencial, especialmente en entornos no estructurados donde la automatización de las acciones del robot puede resultar una tarea muy compleja. En este contexto, el desarrollo de interfaces hombre-máquina (HMI) puede ayudar a incrementar el rendimiento, la versatilidad y la tele operación en el campo de los manipuladores robóticos. Es así que se han desarrollado varias HMI para controlar tanto plataformas robóticas como prótesis mediante el uso de diferentes dispositivos que van desde los más convencionales como teclados y joysticks, hasta sensores comerciales capaces de aprovechar toda la versatilidad del movimiento del ser humano como unidades de medición inercial (IMU) [2], cámaras [3], sensores Kinect [4], entre otros. También se cuentan con sistemas más complejos como los basados en señales electromiográficas (EMG) [2], capaces del reconocimiento de gestos (HGR) [5] que usan principalmente sensores de actividad muscular que usualmente son los de menor costo y de fácil uso, lo que ha ayudado a su investigación y desarrollo.

En tal sentido, en el presente Estudio Técnico, se diseña un sistema conformado por el sistema de reconocimiento de gestos desarrollado en el Proyecto de Investigación PIGR-19-07 basado en el sensor comercial de señales electromiográficas Myo Armband, y el manipulador virtual UR5 de 6 grados de libertad (6 GDL) disponible en el software de simulación CoppeliaSim. Como aplicación de este sistema, se plantea manejar el brazo robótico a través de los gestos clasificados como comandos de selección de modos de funcionamiento, mientras que las señales de la IMU, con la que cuenta el Myo Armband, actuarán como referencia de posición y orientación del efector final del robot virtual UR5. Adicionalmente, para asegurar el seguimiento de trayectorias del manipulador robótico se implementan dos controladores: un PID basado en una solución de mínima norma, y un controlador basado en álgebra lineal. El primero tiene la ventaja de minimizar el esfuerzo de control al encontrar una solución óptima a través de aproximaciones de mínimos cuadrados para alcanzar el objetivo de control, mientras que el segundo, brinda una solución numérica capaz de ser resuelta fácilmente en un software computacional [6].

A continuación, se presentan los objetivos y alcances del presente proyecto. Posteriormente se encuentra una revisión bibliográfica enfocada a los manipuladores robóticos, el software de simulación robótica CoppeliaSim y el sistema de reconocimiento

de gestos desarrollado en el Proyecto de Investigación PIGR-19-07. Luego, se presenta la metodología utilizada tanto para la obtención del modelo cinemático del manipulador robótico, como para el diseño de los controladores, interfaz gráfica y el entorno virtual de CoppeliaSim. Posteriormente, se presentan los resultados del sistema frente a distintos perfiles de referencia que demuestren el funcionamiento del trabajo realizado. Finalmente, las conclusiones y recomendaciones del Proyecto Técnico son presentadas.

1.1 OBJETIVOS

OBJETIVO GENERAL:

Integrar un sistema de reconocimiento de gestos de la mano basado en el sensor MYO Armband con el simulador de robots CoppeliaSim para el control de un manipulador virtual de 6 grados de libertad.

OBJETIVOS ESPECÍFICOS:

- Realizar una revisión bibliográfica referente al manipulador robótico UR5, al uso del simulador de robots CoppeliaSim y sobre el sistema de reconocimiento de gestos de la mano basado en el sensor Myo Armband empleado en el Proyecto de Investigación PIGR-19-07.
- Interconectar el robot virtual UR5 disponible en CoppeliaSim con Matlab para implementar dos algoritmos de control de posición y seguimiento de trayectorias a fin de comandar a este manipulador de 6 grados de libertad.
- Integrar el sistema de reconocimiento de gestos de la mano empleado en el Proyecto de Investigación PIGR-19-07 y basado en el sensor Myo Armband como entrada de referencia de los controladores implementados.
- Implementar una interfaz gráfica en Matlab que permita el monitoreo del robot virtual, del sistema de reconocimiento de gestos y del sistema de control.
- Realizar las pruebas necesarias que permitan verificar el funcionamiento de todo el sistema integrado (robot virtual UR5, controlador y sistema de reconocimiento de gestos).

1.2 ALCANCE

El alcance definido en este Proyecto Técnico se lo describe en los siguientes puntos:

- Se estudia el modelo cinemático del manipulador robótico UR5, lo cual comprende la resolución del problema cinemático directo, inverso y la obtención de su matriz Jacobiana a través de sus parámetros de Denavit-Hartenberg, y de los físicos de su estructura proporcionados por su fabricante.
- Se diseña una escena en el software de simulación robótico CoppeliaSim que incluya el modelo virtual del manipulador robótico UR5 de 6 grados de libertad (6 GDL).
- Se implementa el modelo cinemático del robot manipulador UR5 en Matlab a fin de simularlo y comparar los resultados obtenidos en CoppeliaSim para un mismo patrón de referencia establecido para el movimiento de sus articulaciones.
- Se implementa una interfaz de comunicación entre CoppeliaSim y Matlab a través de una API (Interfaz de Programación de Aplicaciones) que permite la recepción y adquisición de datos entre ambas aplicaciones.
- En base al modelo cinemático, se diseñan e implementan controladores basados en álgebra lineal y tipo PID, los cuales son aplicados al manipulador robótico virtual UR5 en CoppeliaSim para controlar su posición y seguimiento de al menos dos trayectorias predefinidas.
- Se realiza una revisión bibliográfica sobre el uso de señales electromiográficas y del sistema de clasificación de gestos empleado en el Proyecto de Investigación PIGR-19-07 para integrar este sistema de reconocimiento, que emplea el sensor comercial Myo Armband, como entrada de referencia del sistema de control. Para esto, los gestos clasificados serán usados como comandos de selección de modos de funcionamiento del robot, mientras que las señales de la IMU, con la que cuenta el Myo Armband, actuarán como referencia de posición y orientación del efector final del robot virtual UR5.
- Se implementa una interfaz gráfica en Matlab a través de su entorno de desarrollo interactivo App Designer el cual permite monitorear al robot virtual UR5, al reconocimiento de gestos de la mano y se puede visualizar la gráfica del seguimiento de trayectorias predefinidas.
- Se evalúa el funcionamiento del sistema integrado estableciendo una tasa de clasificación en base a los resultados obtenidos con el sistema de reconocimiento de gestos al utilizar los gestos clasificados como comandos de selección de modos de funcionamiento del robot.

- Se evalúa el funcionamiento del sistema integrado, especialmente en cuanto a los sistemas de control diseñados a través del análisis de los errores de posición, velocidad y el uso de criterios de desempeño como el Error Absoluto Integrado (IAE), obtenidos a partir de la simulación de los controladores diseñados de forma independiente al utilizar como entradas de referencia al menos dos trayectorias generadas desde Matlab, cuyos resultados podrán ser visualizados en CoppeliaSim y la interfaz gráfica diseñada en App Designer de Matlab.

1.3 MARCO TEÓRICO

En esta sección se presenta la información teórica necesaria para comprender cada una de las etapas que comprenden el desarrollo de este proyecto técnico.

1.3.1 MANIPULADORES ROBÓTICOS

Los manipuladores robóticos, también conocidos como robots industriales o brazos mecánicos, ver Figura 1.1, son los más comunes y fáciles de reconocer hoy en día, puesto que se encuentran en muchos campos de la industria realizando diferentes tipos de actividades como tareas de ensamblaje, soldadura, pintura, manipulación, e incluso en quirófanos realizando cirugías complejas [1]. La ISO (Organización Internacional de Normalización) los define como: “*manipuladores multifuncionales reprogramables con varios grados de libertad, capaces de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para tareas diversas*” [7].



Figura 1.1 Robots Manipuladores, tomado de [1]

Un robot manipulador está formado por una serie consecutiva de eslabones y articulaciones, donde cada articulación permite conectar dos eslabones consecutivos con el objetivo de formar una cadena cinemática abierta [7]. Este tipo de robot a diferencia de los robots móviles, no son capaces de moverse libremente por el espacio físico que los rodea, sino que poseen una base estática, y un área de trabajo limitada [1].

De forma general, un manipulador robótico está formado por los siguientes elementos:

Articulaciones: Son uniones que permiten la conexión y movimiento rotacional o lineal de traslación entre dos eslabones consecutivos del manipulador. Son básicamente uniones constituidas por servomotores, y pueden ser del tipo lineal o rotacional [7].

Actuadores: Son los elementos encargados de generar movimiento en las articulaciones del manipulador, y pueden ser del tipo eléctrico como servomotores, neumáticos como cilindros de simple o doble efecto, o hasta hidráulicos [7].

Sensores: Proporcionan información del estado interno del robot, que por lo general son variables como posición y velocidad de sus articulaciones, aunque también se utilizan sensores de fuerza o cámaras de video para localizar objetos en su espacio de trabajo lo cual depende principalmente de la actividad que se esté realizando [7].

El espacio de trabajo de un robot manipulador es todo el espacio físico en donde el robot puede realizar sus posibles movimientos y depende principalmente de su geometría y tipos de articulaciones [7].

Sistema mecánico: Son los eslabones rígidos que constituyen la estructura del robot manipulador y se encuentran conectados mediante articulaciones, que por lo general son servomotores. Los eslabones son cuerpos o barras metálicas que se acoplan de forma mecánica a las articulaciones del manipulador [7].

Un robot manipulador se constituye principalmente de un brazo mecánico que asegura su movilidad, una muñeca que le confiere destreza, y un extremo final o también llamado efector final, que corresponde al extremo del último eslabón del robot, destinado a colocar la herramienta más adecuada para una aplicación específica [7].

La posición y orientación del efector final se representa generalmente a través de coordenadas cartesianas y los denominados ángulos de Euler, respectivamente [7].

1.3.1.1 Transformaciones homogéneas

La posición de un manipulador robótico implica describir la posición y orientación de su efector final en un sistema de referencia cartesiano. Por lo tanto, la herramienta matemática que se utiliza para representar operaciones de rotación y traslación son las matrices de transformación homogénea, cuya estructura se utiliza para obtención y presentación del modelo cinemático directo de manipuladores robóticos [8].

Considerando la Figura 1.2, el sistema de referencia fijo $\Sigma_0(x_0, y_0, z_0)$ y el sistema de referencia $\Sigma_1(x_1, y_1, z_1)$, cuyos orígenes no son coincidentes, ya que Σ_1 se encuentra desplazado una distancia d_0^1 con respecto al sistema Σ_0 .

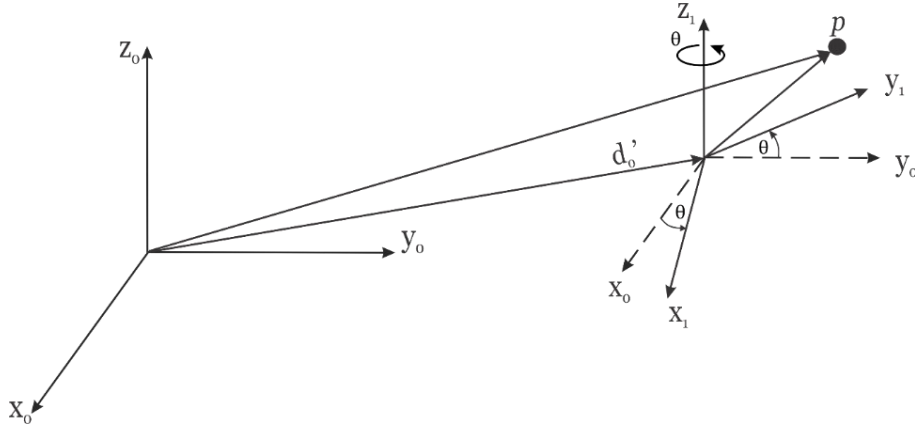


Figura 1.2 Traslación y rotación $R_{z,\theta}$ de los sistemas Σ_0 y Σ_1

La distancia d_0^1 está expresada respecto al sistema de referencia fijo Σ_0 , entonces cualquier vector o punto p puede representarse en coordenadas tanto del sistema fijo Σ_0 como del sistema Σ_1 , por lo que se tendría representación p_0 y p_1 , respectivamente.

Por el ejemplo dado en la Figura 1.2, la relación general entre los sistemas de referencia $\Sigma_0(x_0, y_0, z_0)$ y $\Sigma_1(x_1, y_1, z_1)$ incluye la matriz de rotación $R_{z,\theta}$ y el vector de traslación d_0^1 , mismos que pueden expresarse de la siguiente forma:

$$p_0 = d_0^1 + R_{z,\theta} p_1 \quad (1.1)$$

$$p_0 = \begin{bmatrix} d_{0x}^1 \\ d_{0y}^1 \\ d_{0z}^1 \end{bmatrix} + R_{z,\theta} \begin{bmatrix} p_{1x} \\ p_{1y} \\ p_{1z} \end{bmatrix} \quad (1.2)$$

Si se tiene otro sistema de referencia $\Sigma_2(x_2, y_2, z_2)$ cuya relación con el sistema $\Sigma_1(x_1, y_1, z_1)$ viene dada por una rotación alrededor de su eje x que se representa por una matriz de rotación $R_{x,\phi}$ y una traslación descrita por el vector d_1^2 . Entonces, un punto p_2 del sistema Σ_2 representado en coordenadas del sistema Σ_1 viene dado por:

$$p_1 = d_1^2 + R_{x,\phi} p_2 \quad (1.3)$$

Lo cual expresado respecto al sistema de referencia base Σ_0 viene dado por:

$$p_0 = d_0^1 + R_{z,\theta} [d_1^2 + R_{x,\phi} p_2] \quad (1.4)$$

Por lo tanto, la representación de la Ecuación (1.1) y (1.3), como un conjunto de rotaciones y desplazamientos entre estos sistemas de referencia de forma conjunta y compacta, se conoce como transformación homogénea [7], y está dada por la siguiente expresión:

$$T = \begin{bmatrix} R & d \\ 0^T & 1 \end{bmatrix} \quad (1.5)$$

$$T = \begin{bmatrix} R & d \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} R_{3 \times 3} & d_{3 \times 1} \\ 0_{1 \times 3} & 1_{1 \times 1} \end{bmatrix} \quad (1.6)$$

Esta matriz está compuesta por cuatro submatrices de diferentes dimensiones: una submatriz $R \in SO(3)$ que corresponde a una matriz de rotación; un vector columna $d \in \mathbb{R}^3$ que corresponde al vector de traslación del sistema de referencia Σ_1 ; un vector fila $0_{1 \times 3}$ nulo y un escalar $1_{1 \times 1}$ [8].

1.3.1.2 Cinemática de manipuladores

La cinemática es una rama de la mecánica en donde se estudia y se describe geoméricamente el movimiento de un cuerpo o sistema sin tomar en consideración su masa, y las fuerzas que actúan sobre él [1]. Por esto en Robótica, la cinemática estudia el movimiento del robot manipulador respecto a un sistema de referencia ya que describe de forma analítica su evolución a través del tiempo utilizando la relación existente entre la posición y orientación de su efector final con las posiciones angulares que adoptan sus articulaciones [8]. Básicamente, el estudio de la cinemática de un robot proporciona las herramientas necesarias para analizar y diseñar trayectorias para sí mismo, así como también una orientación adecuada para su herramienta de trabajo [7].

Existen dos problemas a resolver en la cinemática de un manipulador robótico, ver Figura 1.3. El primero se denomina problema cinemático directo, y consiste en determinar cuál es la posición y orientación del efector final del robot, con respecto a un sistema de referencia base conocidos tanto los valores de sus articulaciones, como los parámetros geométricos de los elementos de su estructura. Los valores de sus articulaciones están dados por el vector $q = [q_1, q_2, \dots, q_n]^T \in \mathbb{R}^n$, donde n representa el número de grados de libertad o el número de articulaciones del robot [7]. El segundo, denominado problema cinemático inverso, permite calcular la configuración que deben adoptar cada una de las articulaciones del robot para que el efector final alcance una posición y orientación conocidas [8]. La posición y orientación actual de un robot manipulador puede describirse a partir de 6 coordenadas, 3 representan la posición cartesiana (x, y, z) de su efector final, y 3

coordenadas adicionales, permiten representar la orientación de su herramienta de trabajo, que pueden venir dadas por los denominados ángulos de Euler (θ, ϕ, ψ) [7].

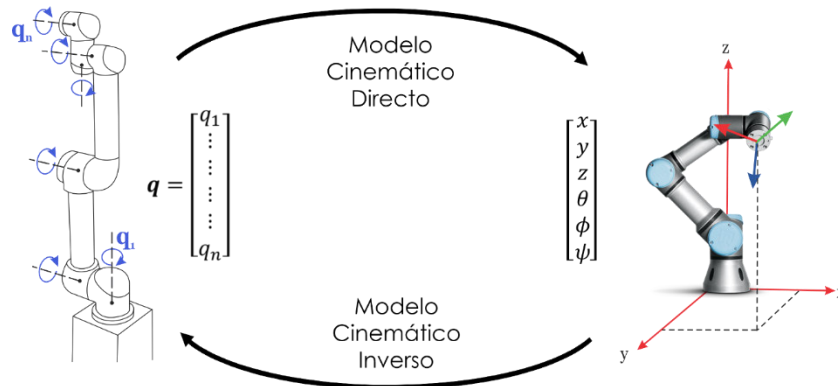


Figura 1.3 Modelo cinemático directo y modelo cinemático inverso

1.3.1.2.1 Modelo Cinemático Directo

El modelo cinemático directo de un manipulador robótico consiste en estudiar de forma analítica su movimiento respecto a un sistema de referencia fijo $\Sigma(x, y, z)$ que relaciona las coordenadas articulares $q \in \mathbb{R}^n$ y las coordenadas cartesianas $[x, y, z]^T \in \mathbb{R}^3$ así como la orientación $[\theta, \phi, \psi]^T \in \mathbb{R}^3$ del efector final del robot, tomando en cuenta las propiedades geométricas de su estructura mecánica. De manera general, el modelo cinemático directo viene dado por la función vectorial:

$$\begin{bmatrix} x \\ y \\ z \\ \theta \\ \phi \\ \psi \end{bmatrix} = f_R(q) \quad (1.7)$$

Donde $f_R(q) \in \mathbb{R}^6$ depende del vector de posiciones articulares $q = [q_1, q_2, \dots, q_n]^T \in \mathbb{R}^n$. Por lo que la resolución de este problema cinemático consiste en encontrar las relaciones que conforman la función vectorial f_R , que permitan conocer cada una de las 6 coordenadas espaciales del efector final del robot, a partir del vector de posiciones articulares $q \in \mathbb{R}^n$. La solución a este problema vendrá dada por las siguientes relaciones [8]:

$$x = f_x(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.8)$$

$$y = f_y(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.9)$$

$$z = f_z(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.10)$$

$$\theta = f_\theta(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.11)$$

$$\phi = f_{\phi}(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.12)$$

$$\psi = f_{\psi}(q_1, q_2, q_3, q_4, \dots, q_n) \quad (1.13)$$

Existen muchos métodos para encontrar este modelo cinemático directo, como, por ejemplo, geometría básica aplicada directamente a la estructura mecánica del robot, pero puede resultar muy compleja si el robot tiene muchos grados de libertad [7]. Otra alternativa ampliamente empleada en la Robótica es el método de Denavit-Hartenberg que brinda un procedimiento sencillo cuyo resultado tiene una estructura basada en transformaciones homogéneas [8].

Método de Denavit-Hartenberg

Este brinda una solución intuitiva para calcular el modelo cinemático directo de manipuladores robóticos con un alto número de grados de libertad. Básicamente, consiste en hallar los valores de una tabla de parámetros relacionados directamente con los eslabones de la estructura del robot [7].

A continuación, se presenta un resumen del procedimiento a seguir para llevar a cabo este método [7]:

1. Localizar la dirección de los ejes z_0, z_1, \dots, z_{n-1} .
2. Establecer el sistema de referencia base $\Sigma_0(x_0, y_0, z_0)$ cuyo origen se coloca en la base del robot. Los ejes x_0 y y_0 son determinados a partir de la regla de la mano derecha. Una vez establecido el sistema $\Sigma_0(x_0, y_0, z_0)$, se inicia un proceso iterativo en donde se ubica el sistema de referencia $\Sigma_i(x_i, y_i, z_i)$ usando el sistema $\Sigma_{i-1}(x_{i-1}, y_{i-1}, z_{i-1})$, iniciando con el sistema de referencia $\Sigma_1(x_1, y_1, z_1)$.

Llevar a cabo los pasos 3 a 5 para las articulaciones $i = 1, \dots, n - 1$.

3. Localizar el origen o_i en la intersección de la normal común que une a los ejes z_i con el eje z_{i-1} . Si el eje z_i intercepta al eje z_{i-1} , colocar el origen o_i en la intercepción. Sin embargo, para el caso en que los ejes z_i y z_{i-1} sean paralelos:
 - Si la i -ésima articulación es rotacional, colocar el origen o_i sobre la articulación i -ésima, tal que $d_i = 0$.
 - Si la i -ésima articulación es prismática, colocar el origen o_i en un punto de límite físico de la articulación i -ésima, por ejemplo, en su extremo final.

4. Seleccionar el eje x_i a lo largo de la normal común que une los ejes z_{i-1} y z_i , cuya dirección se establece desde la articulación $i - 1$ hacia la articulación i .
5. Establecer el eje y_i utilizando la regla de la mano derecha.
6. Establecer el sistema de referencia del efector final del robot $\Sigma_n(x_n, y_n, z_n)$, donde n representa el número de grados de libertad o el número de articulaciones del robot.

Si los sistemas de referencia se asignaron de acuerdo con el procedimiento anterior, se pueden obtener los siguientes parámetros que dependen únicamente de las características geométricas de cada eslabón y articulación, tal y como se muestra en la Figura 1.4 [9]:

7. Obtener q_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.
8. Obtener d_i como la distancia, medida a lo largo del eje z_{i-1} , que habría que desplazar Σ_{i-1} para que x_{i-1} y x_i queden alineados.
9. Obtener a_i como la distancia medida a lo largo del eje x_i que habría que desplazar el nuevo Σ_{i-1} para que su origen coincidiese con Σ_i .
10. Obtener α_i como el ángulo que habría que girar en torno a x_i para que el nuevo Σ_{i-1} coincidiese totalmente con Σ_i .

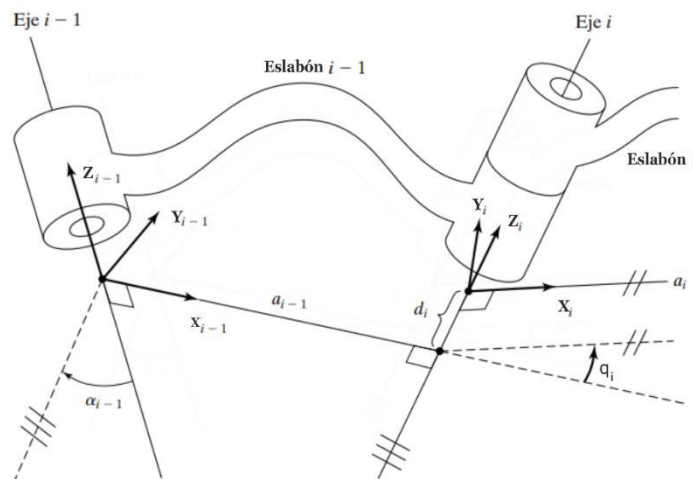


Figura 1.4 Parámetros Denavit-Hartenberg para un eslabón giratorio, tomado de [9]

11. Finalmente, se obtiene las matrices de transformación homogéneas $T_i^{i-1}(q_i)$, donde $i = 1, 2, \dots, n$.

En la representación de Denavit-Hartenberg cada transformación homogénea T_{i-1}^i se representa por el producto de cuatro transformaciones básicas:

$$T_{i-1}^i = R_{z,q} T_{z,d_i} T_{x,a_i} R_{x,\alpha_i} \quad (1.14)$$

$$T_{i-1}^i = \begin{bmatrix} \cos(q_i) & -\cos(\alpha_i) \sin(q_i) & \sin(\alpha_i) \sin(q_i) & a_i \cos(q_i) \\ \sin(q_i) & \cos(\alpha_i) \cos(q_i) & -\sin(\alpha_i) \cos(q_i) & a_i \sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.15)$$

Utilizando la definición de una matriz de transformación, se tiene que la matriz resultante desde la base hasta el efector final de un manipulador robótico de n grados de libertad, puede ser obtenida a partir del siguiente producto:

$$T_0^n = T_0^1 T_1^2 T_2^3 T_3^4 \dots T_{n-1}^n \quad (1.16)$$

$$T_0^n = \begin{bmatrix} \hat{x}_0^n & \hat{x}_0^n & \hat{x}_0^n & P_0^n \\ \hat{y}_0^n & \hat{y}_0^n & \hat{y}_0^n & P_0^n \\ \hat{z}_0^n & \hat{z}_0^n & \hat{z}_0^n & P_0^n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1.17)$$

Esta matriz resultante está compuesta por cuatro submatrices que corresponden a la matriz de rotación del efector final del robot; un vector de traslación del sistema de referencia Σ_n ; un vector fila $0_{1 \times 3}$ nulo y un escalar $1_{1 \times 1}$ [8], tal y como se indica en la Sección 1.3.1.1.

1.3.1.2.2 Modelo Cinemático Inverso

La obtención del modelo cinemático inverso es un problema que consiste en encontrar cada uno de los valores del vector de posiciones articulares $q \in \mathbb{R}^n$ que debe adoptar el manipulador robótico para que su efector final pueda alcanzar una posición con una orientación determinada [8].

$$q = f_{IK}(x, y, z, \theta, \phi, \psi) \quad (1.18)$$

A diferencia de la metodología utilizada en la obtención del modelo cinemático directo, no ocurre lo mismo con el problema cinemático inverso, en donde su resolución consiste en obtención de ecuaciones complejas que están fuertemente ligadas a la configuración del robot. Debido a esto, existen varios métodos para calcular este modelo, algunos consisten en métodos numéricos iterativos, pero su convergencia en tiempo real no está garantizada [8]. Por lo tanto, resulta más adecuado encontrar una relación matemática cerrada que proporcione una solución de forma directa y eficiente. La cual consiste en hallar un número suficiente de relaciones geométricas en donde intervengan las coordenadas del efector

final del robot, sus posiciones articulares y sus dimensiones físicas [8]. Sin embargo, este procedimiento solo es adecuado si el robot posee pocos grados de libertad o sólo se consideren los primeros grados de libertad para posicionarse. La obtención del modelo cinemático inverso del manipulador robótico UR5 se presenta en el siguiente capítulo.

1.3.1.2.3 Matriz Jacobiana

Para conseguir que el efector final de un robot manipulador realice una trayectoria específica con una velocidad constante es necesario, además del modelo cinemático directo, una relación entre las velocidades articulares del robot, y las de la posición y orientación del efector final del robot [8]. Esta relación se consigue a través de la denominada matriz Jacobiana del robot o Jacobiano analítico, que se calcula mediante cinemática diferencial que consiste en la derivada respecto al tiempo de su modelo cinemático directo [8], tal y como se muestra a continuación:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{d}{dt} [x \ y \ z \ \theta \ \phi \ \psi]^T = \frac{d}{dt} f_R(q) \quad (1.19)$$

$$\frac{d}{dt} f_R(q) = \frac{\partial f_R(q)}{\partial q} \dot{q} = J(q) \dot{q} \quad (1.20)$$

Donde $\dot{q} \in \mathbb{R}^n$ es el vector de velocidades articulares del manipulador cuya dimensión depende de su número de articulaciones.

La matriz Jacobiana directa $J(q)$ depende directamente del vector de posiciones articulares $q \in \mathbb{R}^n$, y permite encontrar la velocidad lineal $v = \frac{d}{dt} [x \ y \ z]^T$ y angular $\omega = \frac{d}{dt} [\theta \ \phi \ \psi]^T$ que tiene el efector final del robot conociendo la velocidad de cada una de sus articulaciones $\dot{q} \in \mathbb{R}^n$. A su vez, con su matriz inversa se puede encontrar qué velocidad necesita cada articulación del robot para que su efector final alcance una velocidad dada [8]. Esto se puede ver resumido en la Figura 1.5 para el caso general de un robot de n grados de libertad.



Figura 1.5 Matriz Jacobiana directa y Jacobiana Inversa, tomado de [8]

Adicionalmente, la matriz Jacobiana constituye una de las herramientas más importantes para caracterizar al manipulador, ya que permite analizar singularidades, redundancias, plantear algoritmos de cinemática inversa, y permite inclusive establecer un mapeo entre las fuerzas aplicadas al efector final y los torques de cada articulación [7].

El Jacobiano, a su vez, se divide en dos submatrices; la matriz Jacobiana lineal (J_v) y la de velocidad angular (J_ω). Esto puede representarse de la siguiente manera para un robot de n grados de libertad:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = J(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad (1.21)$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} J_v(q) \\ J_\omega(q) \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad (1.22)$$

La submatriz $J_v(q)$ relaciona la velocidad de las articulaciones del robot con la velocidad lineal de su efector final, mientras que $J_\omega(q)$ la relaciona con su velocidad angular.

El procedimiento necesario para obtener la matriz Jacobiana analítica de un robot manipulador se divide en la obtención de las dos submatrices que forman parte del Jacobiano, cuyo cálculo hace uso del modelo cinemático directo del robot [10]. Entonces, para la matriz Jacobiana de velocidad lineal se consideran las componentes de la última columna de la matriz de transformación homogénea de su modelo cinemático directo, que representa la posición del efector final del robot respecto a su sistema de referencia base. Entonces, para un robot de n grados de libertad se tiene:

$$x = f_x(q_1, q_2, \dots, q_n) \quad (1.23)$$

$$y = f_y(q_1, q_2, \dots, q_n) \quad (1.24)$$

$$z = f_z(q_1, q_2, \dots, q_n) \quad (1.25)$$

A partir de las ecuaciones anteriores, derivando respecto al tiempo ambos lados de cada ecuación, se obtiene:

$$\dot{x} = \sum_1^n \frac{\partial f_x}{\partial q_i} \dot{q}_i \quad (1.26)$$

$$\dot{y} = \sum_1^n \frac{\partial f_y}{\partial q_i} \dot{q}_i \quad (1.27)$$

$$\dot{z} = \sum_1^n \frac{\partial f_z}{\partial q_i} \dot{q}_i \quad (1.28)$$

Expresando de forma matricial se tiene:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = J_v(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad (1.29)$$

De forma que la submatriz Jacobiana de velocidad lineal para un manipulador con n articulaciones rotativas sería de la siguiente forma:

$$J_v = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \frac{\partial f_x}{\partial q_2} & \frac{\partial f_x}{\partial q_3} & \frac{\partial f_x}{\partial q_4} & \dots & \frac{\partial f_x}{\partial q_{n-1}} & \frac{\partial f_x}{\partial q_n} \\ \frac{\partial f_y}{\partial q_1} & \frac{\partial f_y}{\partial q_2} & \frac{\partial f_y}{\partial q_3} & \frac{\partial f_y}{\partial q_4} & \dots & \frac{\partial f_y}{\partial q_{n-1}} & \frac{\partial f_y}{\partial q_n} \\ \frac{\partial f_z}{\partial q_1} & \frac{\partial f_z}{\partial q_2} & \frac{\partial f_z}{\partial q_3} & \frac{\partial f_z}{\partial q_4} & \dots & \frac{\partial f_z}{\partial q_{n-1}} & \frac{\partial f_z}{\partial q_n} \end{bmatrix} \quad (1.30)$$

Al multiplicar esta matriz por el vector de velocidad articular $\dot{q} \in \mathbb{R}^n$ se podrá obtener la velocidad lineal de su efector final respecto al sistema de referencia base del robot.

Para obtener la submatriz de velocidad angular se puede utilizar diferenciación directa del modelo cinemático directo, tal y como se muestra en la Ecuación (1.20), que permite obtener la velocidad angular del efector final dada por la derivada temporal de los ángulos de Euler que hayamos escogido para representar la orientación del efector final del manipulador robótico [7].

$$\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = J_\omega(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad (1.31)$$

Por otro lado, también existe una matriz denominada Jacobiana geométrica, la cual permite obtener los mismos resultados de velocidad lineal del efector final que la Jacobiana analítica, ya que se calcula de forma idéntica, pero a diferencia de esta, su submatriz Jacobiana de velocidad angular permite obtener la velocidad angular del efector final alrededor de los ejes x , y , y z del sistema de referencia base del robot haciendo uso de las matrices de rotación y las contribuciones de velocidad de cada articulación [11]. Por lo que, la velocidad angular del efector final de un robot de n grados de libertad, y las velocidades angulares de cada articulación se relacionan mediante la siguiente expresión [10]:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = R_0^1 \vec{q}_1 + R_0^2 \vec{q}_2 + R_0^3 \vec{q}_3 + R_0^4 \vec{q}_4 + \dots + R_0^n \vec{q}_n = J_\omega \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} \quad (1.32)$$

En donde:

$$\vec{q}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{q}_1 \quad (1.33)$$

Debido a que el ángulo de rotación se produce en torno al eje z de cada articulación. Además, las matrices de rotación se pueden expresar de la siguiente forma:

$$R_0^n = R_0^1 R_1^2 \dots R_{n-1}^n \quad (1.34)$$

Donde R_{i-1}^i es la submatriz de rotación extraída de su respectiva matriz de transformación homogénea T_{i-1}^i .

Entonces, empleando la Ecuación (1.33), se deduce que la tercera columna de cada R_0^i representa la columna i de la matriz Jacobiana directa de velocidad angular (J_ω).

1.3.2 MANIPULADOR ROBÓTICO UR5

El manipulador robótico UR5, mostrado en la Figura 1.6a, es un brazo robótico de 6 grados de libertad (GDL) fabricado por Universal Robots y pertenece a una serie de robots cuya principal característica es su agilidad debido a su peso ligero, velocidad, seguridad, y fácil programación [12]. Todas sus articulaciones son rotacionales, y su posición angular está dada por el vector de posiciones articulares $q = [q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6]^T \in \mathbb{R}^6$, cuya ubicación individual es mostrada en la Figura 1.6b.

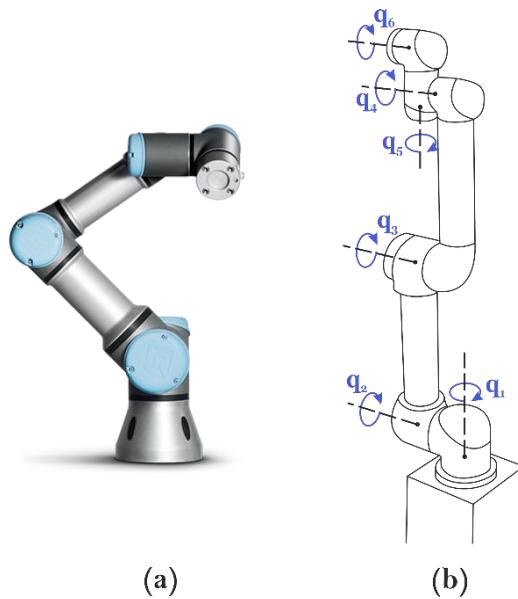


Figura 1.6 Robot UR5 a) Brazo robótico b) Articulaciones rotacionales

Debido a sus ventajas, este robot ha recibido gran atención por parte de la comunidad robótica y la industria, especialmente en la investigación. Sus características técnicas más importantes se resumen en la Tabla 1.1.

Tabla 1.1 Características del Manipulador Robótico UR5 [12]

Característica	Valor
Peso	18.4 kg
Carga Máxima	5 kg
Alcance	850 mm
Grados de libertad	6
Rango de articulaciones	$\pm 2\pi$
Velocidad máxima	π rad/s

1.3.3 CONTROL DE MANIPULADORES ROBÓTICOS

En el presente proyecto se plantean dos técnicas de control para el manipulador robótico UR5, un controlador PID basado en mínima norma y un controlador basado en álgebra lineal, con el objetivo de comparar su desempeño frente a diferentes perfiles de referencia, los cuales están diseñados en base al modelo cinemático del robot, siendo esta su principal ventaja, debido a su simplicidad frente a otras técnicas que hacen uso de su modelo dinámico tales como un controlador por par calculado (CPC) o un controlador de inercia adaptativa (CIA) [2].

1.3.3.1 Control PID de Mínima Norma

El controlador PID es una de las técnicas de lazo cerrado más común y utilizada en el control de procesos automáticos y plataformas robóticas. Esta técnica considera tres esquemas básicos de control con el fin de disminuir o eliminar el error obtenido, que son: control proporcional, control integral y control derivativo. Estos esquemas de control pueden actuar de forma individual o en combinaciones dependiendo de cómo se calibraron sus parámetros en base a la dinámica de cada proceso [13]. La expresión general de un controlador PID viene dada por la siguiente ecuación:

$$U_c(t) = K_d \frac{de(t)}{dx} + K_p e(t) + K_i \int e(t) dt \quad (1.35)$$

Para este proyecto, solo se utiliza la parte proporcional y derivativa, es decir, un control proporcional-derivativo PD que se implementa en base al modelo cinemático del robot, pues es suficiente para que el sistema alcance su estado deseado sin sobrepicos o errores en estado estable sin la necesidad de utilizar la parte integral. La parte proporcional actúa sobre el error de posición, y la parte derivativa actúa sobre el error de velocidad, el cual se obtiene a partir de la diferencia entre la referencia de velocidad y la velocidad del efector final del robot.

En base al esquema de control PD, se ha extendido el mismo a un controlador basado en mínima norma como el presentado en [14]. En este caso, el objetivo de control es que el manipulador realice el seguimiento de trayectorias realizando el menor número posible de movimientos. Por lo que, a continuación, se resume el procedimiento propuesto en [14] para el diseño de dicho controlador.

El modelo cinemático de un robot manipulador viene dado por la siguiente expresión:

$$\dot{h}(t) = \begin{bmatrix} \dot{x}_e(t) \\ \dot{\phi}(t) \end{bmatrix} = J_a(q)\dot{q}(t) \quad (1.36)$$

Donde $\dot{x}_e(t) = [\dot{x}(t) \quad \dot{y}(t) \quad \dot{z}(t)]$ es el vector de velocidades lineales del efector final, $\dot{\phi}(t) = [\dot{\theta}_r(t) \quad \dot{\theta}_p(t) \quad \dot{\theta}_y(t)]$ es el vector que contiene la derivada temporal de los ángulos de Euler que se han escogido para la orientación del efector final, y $\dot{q}(t) = [\dot{q}_1(t) \quad \dot{q}_2(t) \quad \dot{q}_3(t) \quad \dot{q}_4(t) \quad \dot{q}_5(t) \quad \dot{q}_6(t)]$ es el vector de entrada del sistema que posee la velocidad de cada una de las articulaciones del robot manipulador. A continuación, se despeja $\dot{q}(t)$:

$$\dot{q}(t) = J_a^{-1}\dot{h}(t) \quad (1.37)$$

Donde J_a^{-1} es la matriz inversa de la Jacobiana analítica del robot manipulador.

Por lo tanto, la ley de control propuesta es la siguiente:

$$q_c(t) = J_a^{-1} \begin{bmatrix} \dot{x}_d + K_p e_p \\ \dot{\phi}_d + K_o e_o \end{bmatrix} \quad (1.38)$$

Donde, $\dot{x}_d(t)$ y $\dot{\phi}_d(t)$ son los vectores de la derivada temporal de la posición y orientación deseadas del efector final, respectivamente. Además, e_p es el error de posición, y e_o es el error de orientación del efector final. Finalmente, K_p y K_o son matrices definidas positivas que poseen las constantes de calibración del controlador, y se definen de la siguiente forma:

$$K_p = \begin{bmatrix} k_p & 0 & 0 \\ 0 & k_p & 0 \\ 0 & 0 & k_p \end{bmatrix} \quad (1.39)$$

$$K_o = \begin{bmatrix} k_o & 0 & 0 \\ 0 & k_o & 0 \\ 0 & 0 & k_o \end{bmatrix} \quad (1.40)$$

Donde k_p y k_o son constantes positivas.

El error de posición e_p está dado por la siguiente expresión:

$$e_p = p_d - p = \begin{bmatrix} x_d(t) - x(t) \\ y_d(t) - y(t) \\ z_d(t) - z(t) \end{bmatrix} \quad (1.41)$$

Donde p_d y p son la posición deseada y actual del efector final del robot manipulador, respectivamente. Sin embargo, la definición del error de orientación e_o no es tan sencilla, ya que se considera la matriz de rotación actual R_e y deseada R_d del efector final del robot respecto al sistema de referencia base. Por lo tanto, el error de orientación entre R_e y R_d expresado matricialmente está dado por [11]:

$$\tilde{R} = R_d R_e^T \quad (1.42)$$

Sin embargo, el manejo de la matriz \tilde{R} como error de orientación es muy complicado puesto que la ley de control hace uso de la matriz Jacobiana cuyas dimensiones son de 6 x 6 para el caso de un manipulador robótico de 6 grados de libertad. Por lo que se plantea la siguiente definición del error de orientación [15]:

$$e_o(\tilde{R}) = \frac{1}{2} \begin{bmatrix} \tilde{r}_{32} - \tilde{r}_{23} \\ \tilde{r}_{13} - \tilde{r}_{31} \\ \tilde{r}_{21} - \tilde{r}_{12} \end{bmatrix} \quad (1.43)$$

Donde \tilde{r}_{ij} corresponde al ij -ésimo elemento de \tilde{R} , donde $e_o(\tilde{R}) = 0$ siempre y cuando $R_e = R_d$ [15].

1.3.3.2 Controlador basado en Álgebra Lineal

El controlador de álgebra lineal (LAC) proporciona una solución numérica que permite que el robot pase de su estado actual al deseado. Su diseño parte de una aproximación de Euler, basada en su modelo cinemático, por lo que puede ser implementado directamente en un software computacional para manejar una plataforma robótica real. Esto, a diferencia del controlador PID de mínima norma, que al estar en tiempo continuo, se requiere de una técnica de discretización para su implementación [16]. A continuación, se presenta el procedimiento general para su diseño.

El modelo cinemático del manipulador robótico de 6 grados de libertad puede ser reescrito como una aproximación de Euler de la siguiente forma:

$$h_{(k+1)} = h_{(k)} + T_s f(q_{1(k)}, q_{2(k)}, q_{3(k)}, q_{4(k)}, q_{5(k)}, q_{6(k)}) \quad (1.44)$$

Donde:

$$f(q_{1(k)}, q_{2(k)}, q_{3(k)}, q_{4(k)}, q_{5(k)}, q_{6(k)}) = J_{\alpha(k)} \dot{q}_{(k)} \quad (1.45)$$

Donde $\dot{q}_{(k)} = [\dot{q}_{1(k)} \ \dot{q}_{2(k)} \ \dot{q}_{3(k)} \ \dot{q}_{4(k)} \ \dot{q}_{5(k)} \ \dot{q}_{6(k)}]^T$ es el vector de entrada del sistema que posee la velocidad de cada una de las articulaciones del robot manipulador en tiempo discreto, $J_{\alpha(k)}$ es la matriz Jacobiana discreta para cada tiempo de muestreo, T_s es el tiempo de muestreo y $k \in \{0,1,2,3, \dots\}$. Por lo tanto, el modelo cinemático de la Ecuación (1.44) puede ser expresado en tiempo discreto de la siguiente forma:

$$\begin{bmatrix} \frac{x(k+1) - x(k)}{T_s} \\ \frac{y(k+1) - y(k)}{T_s} \\ \frac{z(k+1) - z(k)}{T_s} \\ \frac{\theta_r(k+1) - \theta_r(k)}{T_s} \\ \frac{\theta_p(k+1) - \theta_p(k)}{T_s} \\ \frac{\theta_y(k+1) - \theta_y(k)}{T_s} \end{bmatrix} = J_{a(k)} \dot{q}(k) \quad (1.46)$$

Despejando $\dot{q}(k)$, se tiene:

$$\dot{q}(k) = \frac{J_{a(k)}^{-1}}{T_s} \begin{bmatrix} x(k+1) - x(k) \\ y(k+1) - y(k) \\ z(k+1) - z(k) \\ \theta_r(k+1) - \theta_r(k) \\ \theta_p(k+1) - \theta_p(k) \\ \theta_y(k+1) - \theta_y(k) \end{bmatrix} \quad (1.47)$$

Sin embargo, el modelo presentado en la ecuación anterior también puede ser expresado en términos de la posición y orientación deseada del efector final del robot manipulador de la siguiente forma:

$$\dot{q}(k) = \frac{J_{a(k)}^{-1}}{T_s} \begin{bmatrix} x_d(k+1) - x(k) \\ y_d(k+1) - y(k) \\ z_d(k+1) - z(k) \\ \theta_{dr}(k+1) - \theta_r(k) \\ \theta_{dp}(k+1) - \theta_p(k) \\ \theta_{dy}(k+1) - \theta_y(k) \end{bmatrix} \quad (1.48)$$

Donde se encuentra el cálculo de un error de orientación al igual que en el controlador PID de mínima norma. Por lo que, se utiliza la definición del error de orientación planteada en [15], que considera los ij -ésimos elementos de la matriz de error de orientación obtenida a partir del error entre la matriz de rotación actual R_e y deseada R_d del efector final del robot manipulador.

$$\dot{q}(k) = \frac{J_{a(k)}^{-1}}{T_s} \begin{bmatrix} x_d(k+1) - x(k) \\ y_d(k+1) - y(k) \\ z_d(k+1) - z(k) \\ \tilde{r}_{32}(k+1) - \tilde{r}_{23}(k) \\ \tilde{r}_{13}(k+1) - \tilde{r}_{31}(k) \\ \tilde{r}_{21}(k+1) - \tilde{r}_{12}(k) \end{bmatrix} \quad (1.49)$$

Finalmente, con el fin de mejorar la estrategia de control y manejar la velocidad de la convergencia del error a cero, se introduce un parámetro k para cada una de las coordenadas de posición y orientación, la cual se multiplica directamente con la diferencia entre los valores deseados de posición y orientación con el estado actual del manipulador. Este parámetro puede tomar valores entre 0 y 1, dependiendo de qué tan rápida se necesite que sea la acción de control [13]. Por lo tanto, la ley de control propuesta está definida por la siguiente expresión:

$$\dot{q}_c(k) = \frac{J_a(k)^{-1}}{T_s} \begin{bmatrix} x_{d(k+1)} - k_x(x_{d(k)} - x(k)) - x(k) \\ y_{d(k+1)} - k_y(y_{d(k)} - y(k)) - y(k) \\ z_{d(k+1)} - k_z(z_{d(k)} - z(k)) - z(k) \\ \tilde{r}_{32(k+1)} - k_1(\tilde{r}_{32(k)} - \tilde{r}_{23(k)}) - \tilde{r}_{23(k)} \\ \tilde{r}_{13(k+1)} - k_2(\tilde{r}_{13(k)} - \tilde{r}_{31(k)}) - \tilde{r}_{31(k)} \\ \tilde{r}_{21(k+1)} - k_3(\tilde{r}_{21(k)} - \tilde{r}_{12(k)}) - \tilde{r}_{12(k)} \end{bmatrix} \quad (1.50)$$

Si se requiere una respuesta rápida del sistema, el parámetro k debe adoptar un valor cercano a 0, y si se requiere una respuesta más lenta, se requiere un valor cercano a 1 [13].

1.3.3.3 Rendimiento de los Sistemas de Control

Para evaluar un sistema de control existen diversos criterios de desempeño que permiten evaluar su comportamiento y eficiencia, tales como la Integral del Error Cuadrático (ISE), la Integral del Tiempo multiplicada por el Error Absoluto (ITAE), o el error del sistema en su estado transitorio y permanente, además de que pueden ser utilizados como base para futuras mejoras del sistema del control. Uno de estos criterios es la Integral del Error Absoluto (IAE), que consiste en el cálculo de la integral en el tiempo del error $e(t)$ constituido por el error de posición e_p y orientación e_o , y representa el área bajo la curva de su evolución en el tiempo, por lo que un valor más bajo de este criterio significa un desempeño más eficiente y estable por parte del controlador. Para obtener este criterio se aplica la definición mostrada en la siguiente expresión:

$$IAE = \int |e(t)| dt \quad (1.51)$$

El valor total del criterio se calcula obteniendo la norma euclidiana del vector de errores obtenido de los errores de posición y orientación.

1.3.4 ENTORNO DE SIMULACIÓN COPPELIASIM EDU (V-REP)

CoppeliaSim es un simulador de robots utilizado principalmente en la industria, educación e investigación debido a que su entorno dispone de varios motores de resolución de la dinámica de plataformas robóticas y herramientas que permiten simular diferentes escenarios para estos sistemas, ver Figura 1.7a [17]. Además, cuenta con una versión educativa gratuita que dispone de una amplia librería de robots, sensores, tutoriales y ejemplos para su uso. CoppeliaSim es un software de simulación muy versátil puesto que se puede utilizar de forma independiente o puede ser integrado a una aplicación mucho más compleja, dando libertad al usuario que desee utilizarlo. Esto se debe a que una de sus principales características es la de poseer diferentes tipos de API's (Interfaz de Programación de Aplicaciones) que permiten comunicar la simulación con aplicaciones externas como Matlab, Octave y otras aplicaciones que se basen en C / C++, Python, Lua, Java o Urbi, ver Figura 1.7b. Adicionalmente, al poseer una arquitectura de control distribuida, todos sus elementos y modelos pueden ser manejados de forma individual a través de un script, complementos, nodos, o clientes API remotos como se mencionó anteriormente [17].

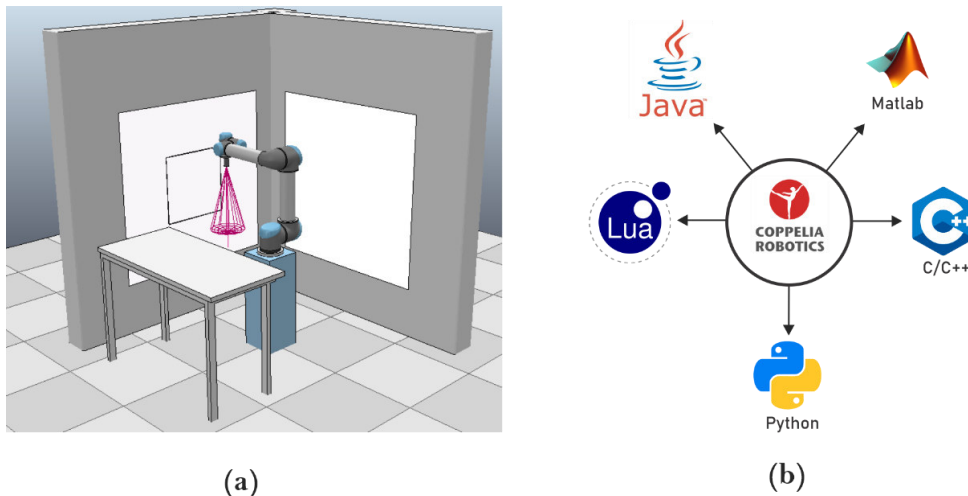


Figura 1.7 Entorno de simulación CoppeliaSim a) Simulación de un manipulador robótico virtual UR5 b) Aplicaciones externas con las que puede interactuar CoppeliaSim

Entre las aplicaciones más importantes que ofrece CoppeliaSim, se tiene:

- Simulación de sistemas automáticos dentro de la industria.
- Monitoreo remoto.
- Control por hardware.
- Desarrollo rápido de algoritmos.

- Educación enfocada a la robótica.

1.3.4.1 API CoppeliaSim

CoppeliaSim posee muchas formas de intercambiar, transmitir y recibir datos o mensajes de forma interna como externa. Para intercambiar datos de forma interna, se utilizan: señales, bloques de datos personalizados, o pequeños scripts. Por otro lado, para intercambiar información con aplicaciones externas, otras computadoras o incluso otras máquinas se utilizan mensajes ZMQ, interfaces ROS, API's remotas, interfaces BlueZero, puertos seriales y sockets [18].

Para este proyecto, se utiliza su API remota, la cual permite controlar la simulación de CoppeliaSim con una aplicación o hardware externo, que, en este caso, es Matlab. Esto se debe a que posee una cantidad de aproximadamente 100 funciones API remotas para interactuar con la simulación, y puede ser utilizada con aplicaciones basadas en C / C++, scripts de Python, Lua, aplicaciones Java o programas de Matlab/Octave [17].

1.3.5 SISTEMAS DE RECONOCIMIENTO DE GESTOS

Un sistema de reconocimiento de gestos es básicamente una interfaz hombre-máquina cuyo objetivo es determinar en qué momento y qué gesto fue realizado [5]. Los gestos pueden ser resultado de algún movimiento o estado de una parte del cuerpo, que por lo general pueden ser del rostro o mano.

Una de las etapas más importantes de un sistema de reconocimiento es la adquisición de datos, que consiste en medir diferentes señales generadas producto de la actividad muscular cuando un usuario realiza un movimiento. Para la adquisición de señales se pueden utilizar diferentes tipos de dispositivos como sensores de visión [3], [19], unidades de medición inercial (IMU) y dispositivos basados en electromiografía (EMG) [2], [19]. Algunos sistemas también combinan diferentes tipos de señales de varios sensores para mejorar la precisión del sistema [19]. Adicionalmente, también existen dispositivos EMG comerciales como el dispositivo Myo Armband que pueden medir la actividad eléctrica de los músculos del antebrazo.

Los sistemas de reconocimiento de gestos se utilizan tanto en aplicaciones industriales como en domésticas. Estos sistemas detectan el instante en que se realiza un gesto e identifican su clase a partir de un conjunto predefinido. Por lo que, la información que brindan se puede utilizar como una herramienta importante para utilizarse dentro de interfaces hombre-máquinas (HMI) que permitan controlar plataformas robóticas.

Varios trabajos han utilizado previamente EMG para aplicaciones que emplean plataformas robóticas. Existen muchos trabajos en donde se desarrollaron interfaces hombre-máquina (HMI) para controlar diferentes tipos de robots mediante el uso de señales EMG, IMU's y sensores de visión. Por ejemplo, en [19], se presenta un sistema que usa la estimación del movimiento de las extremidades superiores basada solo en mediciones EMG y un sensor Kinect para controlar un brazo robótico humanoide móvil con movimientos aleatorios del brazo y perfiles de velocidad variables de la mano. En [20], se diseñó una HMI que usaba la orientación del antebrazo, la fuerza muscular y los movimientos dinámicos de la mano para controlar un manipulador robótico de seis grados de libertad (GDL) con una pinza de un grado de libertad a través de un conjunto de sensores EMG y una unidad de medición inercial. El sistema propuesto fue capaz de reconocer movimientos dinámicos de la mano que pueden cambiar poses y configuraciones en tiempo real. En [21], se presentó un proyecto en donde se desarrolló una HMI para controlar un robot submarino utilizando señales EMG y otras medidas proporcionadas por un giroscopio y una cámara. Utilizaron un algoritmo fuzzy-PID para controlar el robot submarino para alcanzar diferentes posturas. Finalmente, en [2] se desarrolló un control robusto no lineal para resolver el seguimiento de trayectorias de un manipulador robótico de tres grados de libertad (GDL). En este caso, los autores diseñaron una HMI utilizando dos dispositivos Myo Armband y un sistema de reconocimiento de gestos para rastrear el movimiento de ambos brazos.

1.3.5.1 Señales Electromiográficas

La electromiografía, o también conocida de forma abreviada como EMG, es un proceso en donde se estudia la actividad eléctrica de un músculo del cuerpo, utilizando un electrodo que necesita una referencia, un punto común o tierra y un método de almacenamiento [22]. Esta actividad eléctrica, se representa a partir de señales electromiográficas o EMG, que matemáticamente se pueden representar a partir de la sumatoria de todas las acciones de potencial de las unidades motoras (MUAPs, por sus siglas en inglés) del músculo bajo análisis [23], a partir de la siguiente expresión:

$$EMG(t) = \sum_{k=1}^N u_n(t) \quad (1.52)$$

Donde $u_n(t)$ es el tren de acciones de potencial de cada unidad motora.

Una acción de potencial de una unidad motora es un pulso eléctrico que viaja por las fibras musculares, las cuales son las encargadas de brindar fuerza a los músculos para que estos sean capaces de moverse. Cada unidad motora recibe órdenes de una única motoneurona,

las cuales sirven de interconexión entre la médula espinal y las fibras musculares, ver Figura 1.8a. Finalmente, en la Figura 1.8b, se muestra cómo funciona una MUAP, que en conjunto se denomina tren de acciones de potencial de una unidad motora (MUAPT, por sus siglas en inglés) [22].

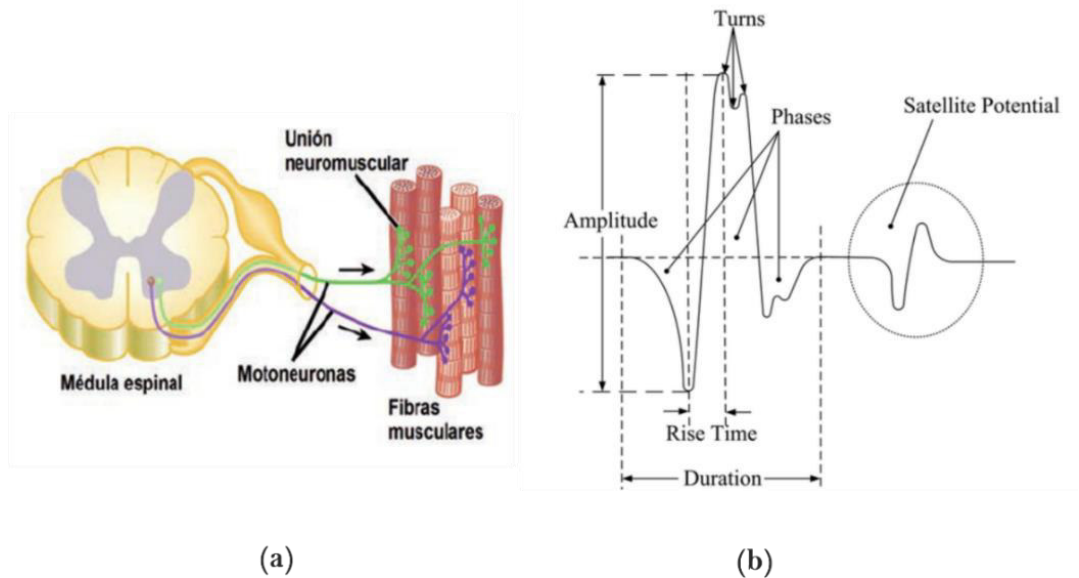


Figura 1.8 Señales EMG a) Unidad motora de un músculo b) Acción de potencial de una unidad motora (MUAP), tomado de [24] y [25]

Para el caso del presente proyecto, el sistema de reconocimiento de gestos utilizado en este proyecto adquiere las señales EMG a través del sensor Myo Armband y utiliza modelos de aprendizaje computacional o *machine learning* que permiten discriminar los diferentes gestos a partir de un conjunto de datos y sus etiquetas.

1.3.5.2 Sensor Myo Armband

El sensor Myo Armband, ver Figura 1.9, es un dispositivo electrónico que permite medir y entregar las señales electromiográficas superficiales (EMGs) de sus diferentes sensores (electrodos) cuando éste se coloca en el antebrazo de un usuario, y es distribuido en el mercado por la empresa Thalmic Labs Inc. Adicionalmente, también es posible obtener datos de la IMU (Unidad de Medición Inercial) provista en el sensor Myo, esto incluye: la representación de orientación mediante cuaternios, medición de la velocidad angular en los tres ejes principales mediante giroscopios y medición de aceleración lineal en los tres ejes principales mediante acelerómetros [26].



Figura 1.9 Sensor Myo Armband™ por Thalmic Labs Inc, tomado de [26]

1.3.5.2.1 **Características del sensor Myo Armband**

Las principales características del Myo Armband son [26]:

- IMU (acelerómetro, giroscopio y magnetómetro en los ejes x, y, z).
- Frecuencia de muestreo de señales EMG: 200 Hz.
- Frecuencia de muestro de la unidad de medición inercial (IMU): 50 Hz.
- Comunicación Bluetooth.
- Batería recargable de ion litio de larga duración.
- Retroalimentación háptica a través de vibraciones.

1.3.5.3 **Sistema Desarrollado en el Proyecto de Investigación PIGR-19-07**

El presente trabajo de titulación se realiza dentro del marco del Proyecto de Investigación Grupal PIGR-19-07 “Reconocimiento de gestos de la mano usando señales electromiográficas e inteligencia artificial y su aplicación para la implementación de interfaces humano - máquina y humano - humano”, proyecto en el cual se ha desarrollado un sistema de reconocimiento de gestos de la mano. Este sistema de reconocimiento utilizó el dispositivo Myo Armband y se implementó completamente en Matlab. Es capaz de clasificar y reconocer seis gestos de la mano diferentes. En la Figura 1.10, se muestran cada de uno de los gestos.

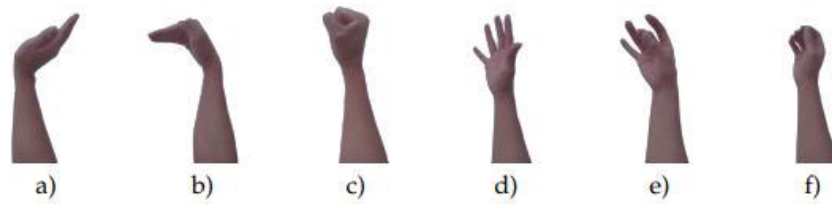


Figura 1.10 Gestos de la mano a ser reconocidos por el sistema HGR a) WAVE IN b) WAVE OUT c) FIST d) OPEN e) PINCH f) RELAX, tomado de [5]

WAVE IN: Mano en ángulo de 90° hacia adentro, y al terminar el gesto se regresa a la posición de reposo o relax.

WAVE OUT: Mano en ángulo de 90° hacia afuera, y al terminar el gesto se regresa a la posición inicial o reposo.

FIST: Se cierra la mano en forma de puño con un poco de fuerza, y después de un corto tiempo se regresa a la posición de reposo.

PINCH: Se realiza un golpe rápido o pellizco entre los dedos pulgar y medio, y al terminar el gesto se regresa a la posición de reposo.

OPEN: Se mantiene la mano abierta con los dedos extendidos. Para terminar el gesto se regresa a la posición inicial o de reposo.

RELAX: Se mantiene con el antebrazo relajado y la mano sin movimiento, también se conoce como “Nogesture” o posición de reposo.

El sistema HGR utilizado se divide en dos tipos: el modelo general y el modelo específico. El modelo específico requiere un entrenamiento previo por cada vez que un nuevo usuario requiera utilizarlo. Por otro lado, el modelo general no necesita de un entrenamiento previo para utilizarse, ya que posee una amplia base de muestras (data set) de múltiples usuarios. El HGR en cuanto a reconocimiento, mostró un rendimiento de hasta el 81,3% para el modelo general, y un rendimiento 94.2% para el específico [5].

Este sistema de reconocimiento se compone de cinco etapas, que son la adquisición de datos, el preprocesamiento, la extracción de características, la clasificación y el posprocesamiento. En la Figura 1.11, se muestran cada una de las etapas mencionadas formando parte de la arquitectura general de un sistema de reconocimiento de gestos.

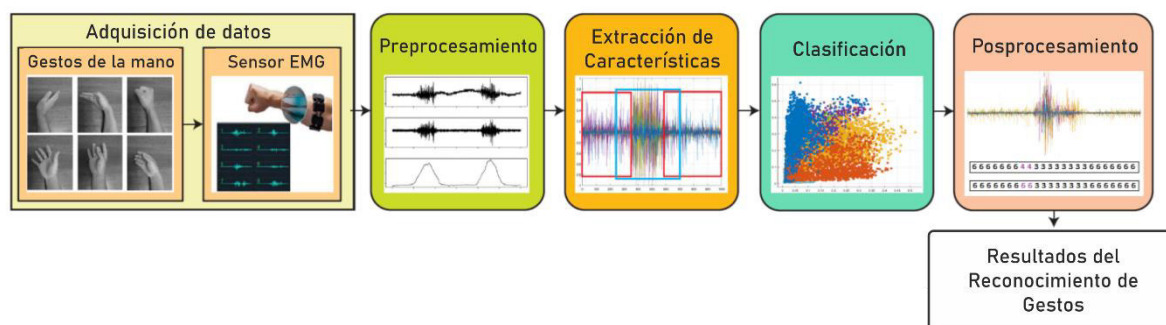


Figura 1.11 Arquitectura de un sistema de reconocimiento de gestos de la mano, tomado de [5]

1.3.5.3.1 Adquisición de datos

La adquisición de datos consiste en la colocación del sensor Myo Armband en el antebrazo del usuario para obtener sus señales EMG. Éstas son adquiridas por cada uno de sus ocho electrodos diferenciales que trabajan a una frecuencia de muestreo de 200 Hz. El dispositivo Myo Armband transmite los datos recopilados vía Bluetooth a una computadora para ser utilizados dentro de Matlab, por lo que se hace uso de la librería “Myo Mex” de código abierto disponible en Github, desarrollada por Mark Tomaszewski [27].

La distribución de electrodos del dispositivo Myo Armband y la posición sugerida por el fabricante se muestran en la Figura 1.12a y Figura 1.12b, respectivamente.

Adicionalmente, en la Figura 1.12c, se muestra el Myo Armband con una posición diferente a la recomendada por el fabricante.

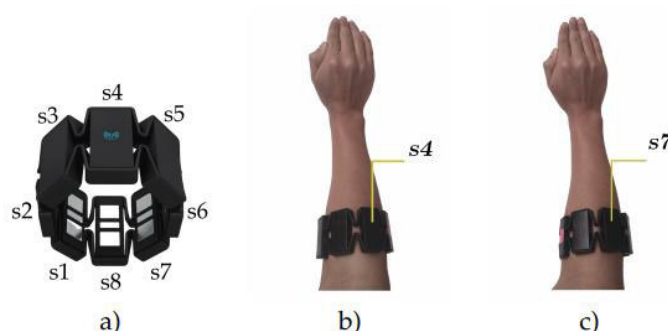


Figura 1.12 Sensor Myo Armband a) Distribución de sensores b) posición de sensores sugerida por el fabricante c) posición del Myo Armband rotada que podrían causar problemas durante el reconocimiento, tomado de [5]

Este sistema HGR utiliza el conjunto de datos o *data set* recopilado en una investigación anterior y disponibles en [28]. Este conjunto de datos está compuesto por 612 usuarios y se dividió en dos grupos: 50% para capacitación y 50% para pruebas (es decir, 306

usuarios para cada uno). Cabe señalar que el conjunto está compuesto por un 96% de personas diestras y un 4% de personas zurdas, así como un 66% de hombres y 34% mujeres. Además, la distribución por edad tiene una mayor concentración de usuarios entre 18 y 25 años [5].

1.3.5.3.2 Preprocesamiento

La etapa de preprocesamiento consiste en seleccionar la información más importante de las señales EMG para el sistema de reconocimiento, es decir, se realiza un proceso de acondicionamiento o eliminación de componentes innecesarias que dificulten el procesamiento y extracción de características de las señales eléctricas.

Se utiliza una técnica de inventanado de una señal, que consiste en la selección de un intervalo de la señal eléctrica, para posteriormente identificar qué porción de la señal EMG necesita ser procesada y clasificada cuando ésta sobrepase un límite de energía, la cual puede ser calculada a través de la siguiente expresión [5]:

$$E = \sum_{i=1}^L abs\{(x_i) \cdot abs(x_i) - (x_{i-1}) \cdot abs(x_{i-1})\} \quad (1.53)$$

Donde x_i es una muestra de la señal EMG, L es el número total de datos de la señal EMG y abs corresponde al valor absoluto.

1.3.5.3.3 Extracción de características

La extracción de características es un proceso en donde los atributos de las señales EMG son calculados y organizados en un vector. En este sistema HGR, se utilizan cinco funciones encargadas de la extracción de características que representen información útil y significativa de la señal EMG, con el fin de aumentar el desempeño de la etapa de clasificación. Este conjunto de funciones se explica brevemente a continuación:

Desviación estándar: Esta función mide la dispersión de los datos que conforman la señal EMG. Es decir, indica qué tan dispersos se encuentran los datos del promedio. Y se puede expresar de la siguiente forma:

$$SD = \sqrt{\frac{1}{L-1} \sum_{i=1}^L |x_i - u|^2} \quad (1.54)$$

Donde x_i es una muestra de la señal EMG, u es el valor promedio, y L es el número total de datos de la señal EMG.

Envolvente absoluta: Utiliza la transformada de Hilbert [29] para calcular los atributos instantáneos de una función a través del tiempo, especialmente amplitud y frecuencia.

$$AE = |AE| = \sqrt{f(t)^2 + (H\{f(t)\})^2} \quad (1.55)$$

Donde $H(t)$ es la transformada de Hilbert y $f(t)$ es la señal EMG.

Valor medio absoluto: Este parámetro indica el promedio del valor absoluto de la amplitud de la señal EMG, y se define por la siguiente ecuación:

$$MAV = \frac{1}{L} \sum_{i=1}^L x_i \quad (1.56)$$

Donde x_i es una muestra de la señal EMG y L es el número total de datos de la señal EMG.

Energía: Es un parámetro que permite medir la distribución de energía de la señal EMG, y se puede calcular a partir de la Ecuación (1.53).

Raíz del error cuadrático medio: representa matemáticamente la fuerza y contracción del músculo al momento de realizar los diferentes gestos, y se define de la siguiente forma:

$$RMS = \sqrt{\frac{1}{L} \sum_{i=1}^L (x_i)^2} \quad (1.57)$$

Donde x_i es una muestra de la señal EMG y L es el número total de datos de la señal EMG.

1.3.5.3.4 Clasificación

Un sistema de reconocimiento requiere la implementación de un clasificador que pueda identificar, separar o asignar a qué clase corresponde cada una de las señales EMG o gestos realizados dentro de un conjunto determinado de clases. En el sistema HGR desarrollado en el Proyecto de Investigación Grupal PIGR-19-07 se utiliza la técnica que se denomina *support vector machine* (SVM), implementada en Matlab, que utiliza aprendizaje computacional o *machine learning*, debido a su rendimiento y bajo costo computacional, siendo capaz hasta de superar a la técnica *k-nearest neighbor* (KNN) en sistemas de clasificación [5].

La técnica SVM utiliza una función de kernel polinómica de tercer grado para llevar a cabo la etapa de clasificación a partir de sus datos de entrada, que se reasignan en un nuevo hiperplano que facilita la separación de clases [5]. En esta técnica, se utiliza clasificación multiclase o *multi-class classification* desglosando el problema de clasificación múltiple a

varios casos de clasificación binaria, que se denominan *one-vs.-one coding* [5]. El entrenamiento de esta técnica se realizó de forma offline, obteniendo diferentes conjuntos de vectores de soporte, tanto para el modelo específico, como para el general. Finalmente, la técnica SVM al clasificar una ventana de señal EMG, entrega una etiqueta del gesto clasificado junto a una matriz de puntajes de cada gesto, cuyos valores numéricos se analizan para determinar si la etiqueta del gesto predicha es válida o se considera como noGesture o posición de reposo [5].

1.3.5.3.5 Posprocesamiento

Durante la etapa de clasificación, cada ventana deslizante para analizar la señal EMG es de 200 puntos con 20 puntos de separación entre sí, posteriormente se obtiene un vector con la probabilidad por cada clase de gesto, y sólo se considera la clase más probable como resultado de la etapa de clasificación.

En esta etapa, se reciben estos resultados y un vector de etiquetas es creado, concatenándolos. Este vector se crea una vez que el número de ventanas analizadas alcancen 5 segundos de grabación. Posteriormente, se analiza cada grupo de 4 etiquetas, cuyos resultados se almacenan en nuevo vector B^* , cuyo resultado es clave para remover posibles falsos positivos de las diferentes clases de gestos. Estos se asignan a cada punto del dominio del tiempo dependiendo de la posición de cada ventana deslizante, tal y como se muestra en la Figura 1.13 [5]. Adicionalmente, también se utiliza el *ground truth* A^* , un vector booleano establecido sobre la actividad muscular, que es incluido en cada muestra del *data set*, que se obtiene a partir de segmentación manual [5].

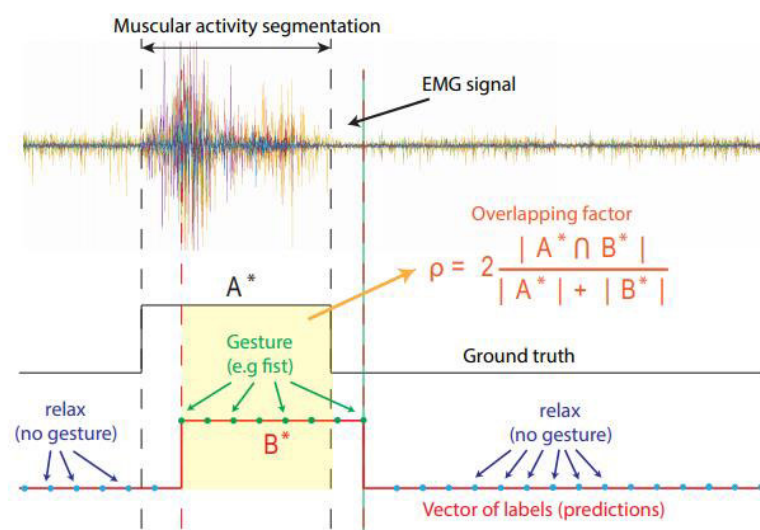


Figura 1.13 Etapa de posprocesamiento del sistema HGR, tomado de [5]

Finalmente, el reconocimiento es considerado exitoso si el vector de etiquetas corresponde con las del *ground truth*, y si estas etiquetas se alinean con la segmentación manual del *ground truth* en el dominio del tiempo [5]. Por lo tanto, se utiliza un factor de solapamiento mínimo de $\rho = 0.25$ como umbral para decidir si el reconocimiento es o no correcto, y se define a partir de la siguiente ecuación [5]:

$$\rho = 2 \frac{|A^* \cap B^*|}{|A^*| + |B^*|} \quad (1.58)$$

2 METODOLOGÍA

2.1 ARQUITECTURA DEL SISTEMA

En la Figura 2.1 se presenta la integración del sistema de reconocimiento de gestos, desarrollado en el Proyecto de Investigación PIGR-19-07, y la unidad de medición inercial (IMU) del sensor Myo Armband con un entorno virtual desarrollado en CoppeliaSim, con el objetivo de controlar un manipulador robótico UR5 virtual de 6 grados de libertad. Adicionalmente, se desarrolla una interfaz gráfica en App Designer de Matlab que permite al usuario interactuar con la simulación del robot virtual, monitorear su estado, elegir su controlador, y escoger su perfil de referencia.

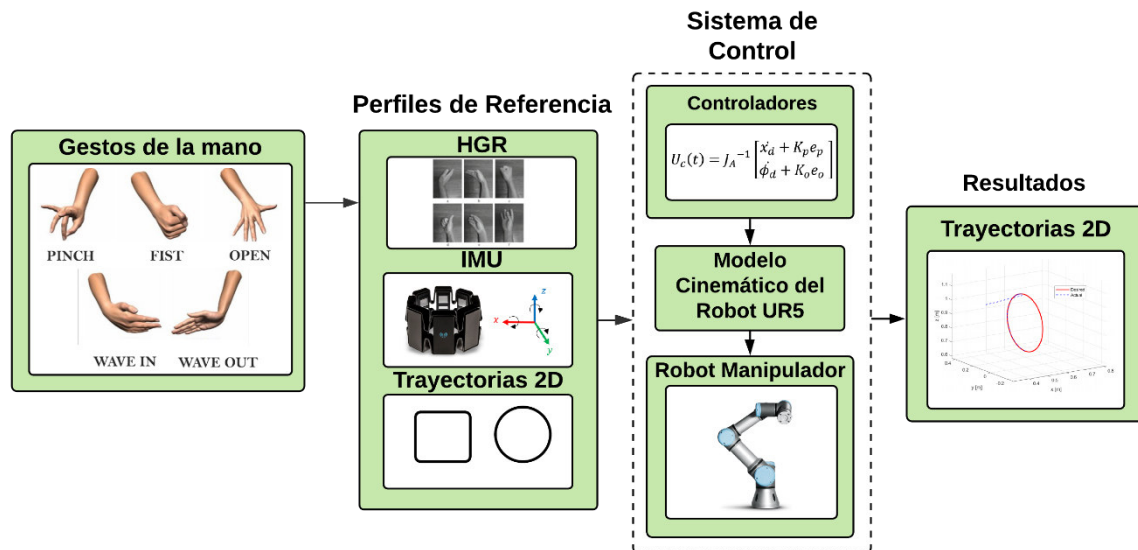


Figura 2.1 Arquitectura de la interfaz humano-máquina propuesta

El sistema presentado en la Figura 2.1 posee dos bloques principales, los perfiles de referencia y el sistema de control. En los perfiles de referencia se tiene el sistema de reconocimiento de gestos de la mano, la IMU del sensor Myo Armband y las trayectorias generadas a través de comandos básicos de Matlab. En la parte del sistema de control, se tiene el modelo cinemático directo del robot, su matriz Jacobiana inversa, sus controladores, y finalmente, su modelo virtual, disponible en CoppeliaSim, que forma parte de un entorno virtual diseñado para ejecutar sus pruebas de funcionamiento. A continuación, se inicia detallando lo referente a los componentes del bloque del Sistema de Control y posteriormente se explicará su integración con el sistema de reconocimiento de gestos a través del bloque de Perfiles de Referencia.

2.2 MODELO CINEMÁTICO DEL BRAZO ROBÓTICO UR5

Un robot manipulador se puede representar mediante un modelo cinemático que estudia su movimiento sin analizar las fuerzas que lo provocan, o a través de un modelo dinámico que estudia las fuerzas y momentos que originan ese movimiento. Sin embargo, la identificación del modelo dinámico de un robot es generalmente más complejo ya que depende de la estructura mecánica del robot, lo que complica el desarrollo de un sistema de control [12]. Por esta razón, muchos controladores se diseñan tomando en consideración principalmente el modelo cinemático, obteniendo resultados aceptables para velocidades bajas y cambios de trayectoria suaves [8]. Por lo que se puede modelar utilizando el método de Denavit-Hartenberg (DH) y las propiedades geométricas de su estructura [7], como se explicó en la Sección 1.3.1.2.1.

2.2.1 MODELO CINEMÁTICO DIRECTO

En la Figura 2.2, se ilustra el esquema del manipulador robótico UR5 y la ubicación de sus sistemas de referencia para cada una de sus articulaciones a través del método de Denavit-Hartenberg. Los parámetros y matrices de transformación utilizadas para el modelo cinemático se basan en la ubicación de estos sistemas de referencia.

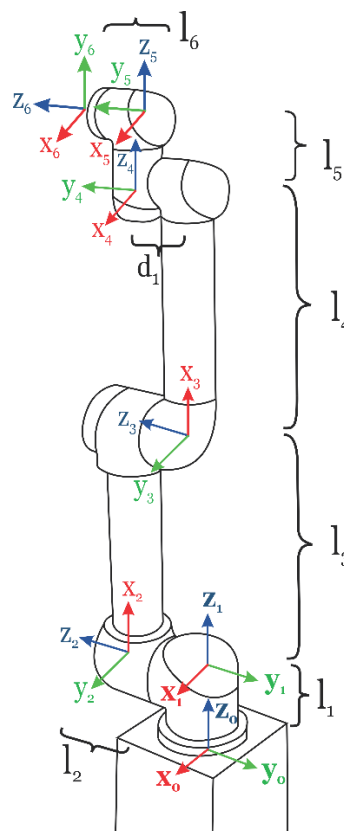


Figura 2.2 Articulaciones y ubicación de los sistemas de referencia en el Robot UR5

Los parámetros geométricos de la estructura del robot manipulador UR5, presentados en la Figura 2.2, son indispensables para el funcionamiento tanto del modelo cinemático como de sus controladores, y en consecuencia del sistema, por lo tanto, en la Tabla 2.1 se presentan estas medidas.

Tabla 2.1 Parámetros geométricos del Manipulador Robótico UR5 [12]

Parámetro	Valor [m]
l1	0.089
l2	0.10
l3	0.42
l4	0.39
l5	0.095
l6	0.082
d1	0.110

Los parámetros de Denavit-Hartenberg (DH) del manipulador robótico UR5 para la configuración que se muestra en la Figura 2.2 se presentan en la Tabla 2.2.

Tabla 2.2 Parámetros DH del Robot UR5

i	a_i	α_i	d_i	q_i
1	0	0	l_1	q_1
2	0	$\pi/2$	l_2	q_2
3	l_3	0	$-l_2$	q_3
4	l_4	0	l_2	q_4
5	0	$-\pi/2$	l_5	q_5
6	0	$\pi/2$	0	q_6

Utilizando la definición de matriz de transformación T_{i-1}^i , presentada en la Ecuación (1.15), se conoce que la cinemática directa de la posición y orientación del robot puede ser obtenida a partir del siguiente producto:

$$T_0^6 = T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 \quad (2.1)$$

$$T_0^6 = \begin{bmatrix} \hat{x}_{0_x}^6 & \hat{x}_{0_x}^6 & \hat{x}_{0_x}^6 & P_{0_x}^6 \\ \hat{y}_{0_y}^6 & \hat{y}_{0_y}^6 & \hat{y}_{0_y}^6 & P_{0_y}^6 \\ \hat{z}_{0_z}^6 & \hat{z}_{0_z}^6 & \hat{z}_{0_z}^6 & P_{0_z}^6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

Desarrollando, se tiene principalmente las coordenadas de posición de su efector final, que están dadas por las siguientes ecuaciones:

$$\begin{aligned}
P_{0x}^6 &= l_2 s_1 - l_3 c_1 s_2 + l_6 c_5 s_1 - l_4 c_1 c_2 s_3 - l_4 c_1 c_3 s_2 - l_5 c_1 c_2 c_4 s_3 - l_5 c_1 c_3 c_4 s_2 + l_5 c_1 c_3 s_3 s_4 \\
&\quad + l_6 c_1 c_2 c_3 c_4 s_5 - l_6 c_1 c_2 s_3 s_4 s_5 - l_6 c_1 c_3 s_2 s_4 s_5 - l_6 c_1 c_4 s_2 s_3 s_5 \\
P_{0y}^6 &= l_5 s_1 s_2 s_3 s_4 - l_6 c_1 c_5 - l_3 s_1 s_2 - l_4 c_2 s_1 s_3 - l_4 c_3 s_1 s_2 - l_5 c_2 c_3 s_1 s_4 - l_5 c_2 c_4 s_1 s_3 \\
&\quad - l_5 c_3 c_4 s_1 s_2 - l_2 c_1 + l_6 c_2 c_3 c_4 s_1 s_5 - l_6 c_2 s_1 s_3 s_4 s_5 - l_6 c_3 s_1 s_2 s_4 s_5 \\
&\quad - l_6 c_4 s_1 s_2 s_3 s_5 \\
P_{0z}^6 &= l_1 + l_4 c_{23} + (l_6 c_{234-5})/2 + l_3 c_2 + l_5 c_{234} - (l_6 c_{2345})/2
\end{aligned} \tag{2.3}$$

Donde s_{234} representa $\sin(q_2 + q_3 + q_4)$ y c_{234} el $\cos(q_2 + q_3 + q_4)$.

En el Anexo A, se presenta el modelo cinemático directo desarrollado completamente, que incluye tanto la matriz de rotación del efector final, como su vector de posición respecto al sistema coordenado 0.

2.2.2 MODELO CINEMÁTICO INVERSO

Una vez establecidos los sistemas de referencia en base el método de Denavit-Hartenberg, se procede a obtener las expresiones que permitan calcular el modelo cinemático inverso del manipulador robótico UR5 mediante un análisis geométrico de su estructura.

Cálculo de q_1

Siendo T_6^0 y l_6 conocidos, se determina la localización del sistema de referencia 5 en relación con el sistema de referencia base del robot. Entonces, para cualquier punto dentro del área de trabajo del manipulador, P_0^5 corresponde a la localización de sistema de referencia 5, y se obtiene a partir de la traslación de $-l_6$ unidades en el eje z del del sistema de referencia 6, tal y como se muestra en la Figura 2.3.

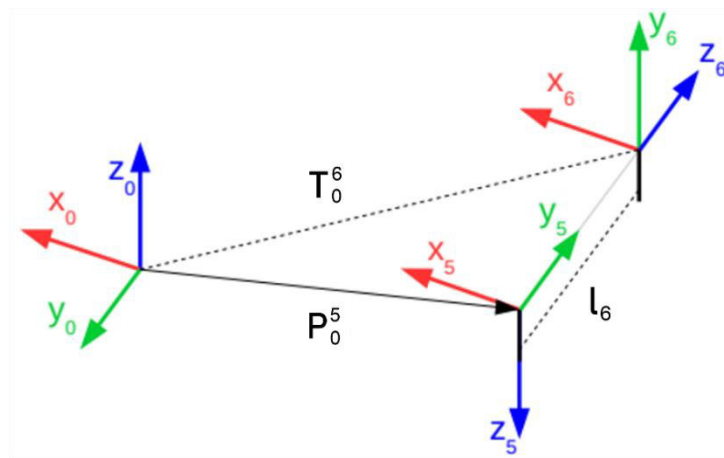


Figura 2.3 Origen del sistema de referencia 5 respecto al sistema de referencia base, tomado de [30]

Entonces, el origen del frame P_5^0 puede ser descrito por la siguiente ecuación:

$$P_0^5 = T_0^6 \begin{bmatrix} 0 \\ 0 \\ -l_6 \\ 1 \end{bmatrix} \quad (2.4)$$

Entonces, conocida la posición P_0^5 , ver Figura 2.4, se puede realizar el siguiente análisis:

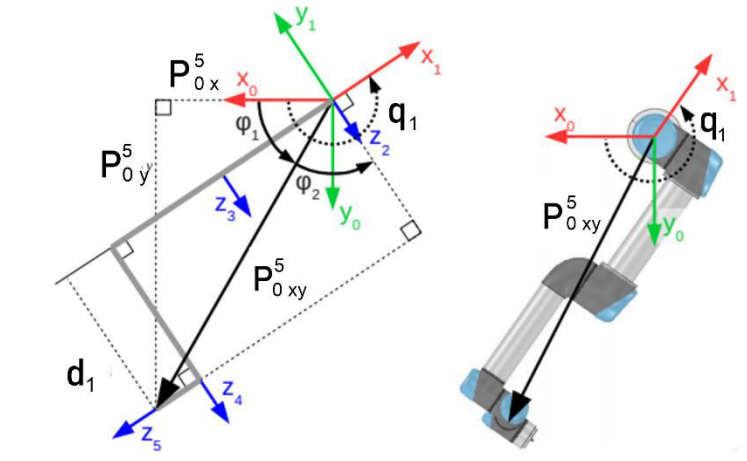


Figura 2.4 Análisis geométrico de P_0^5 , tomado de [30]

A partir de la figura anterior, se tiene que el ángulo q_1 es igual a:

$$q_1 = \varphi_1 + \varphi_2 + \frac{\pi}{2} \quad (2.5)$$

En donde, el ángulo φ_1 se obtiene de forma geométrica a través del triángulo rectángulo formado por los catetos $P_0^5_y$ y $P_0^5_x$, y es igual a:

$$\varphi_1 = \text{atan2}(P_0^5_y, P_0^5_x) \quad (2.6)$$

Y el ángulo φ_2 se obtiene a través del triángulo formado por $|P_0^5_{xy}|$ y D_1 de la siguiente forma:

$$\cos(\varphi_2) = \frac{D_1}{|P_0^5_{xy}|} \quad (2.7)$$

$$\varphi_2 = \pm \text{acos} \left(\frac{D_1}{|P_0^5_{xy}|} \right) \quad (2.8)$$

Entonces, el ángulo q_1 es igual a:

$$q_1 = \text{atan2}(P_{0y}^5, P_{0x}^5) \pm \arccos\left(\frac{D_1}{|P_{0xy}^5|}\right) + \frac{\pi}{2} \quad (2.9)$$

En la ecuación anterior se tiene la función coseno inversa, por lo que q_1 tiene dos posibles respuestas, y corresponden a cuando el manipulador se encuentra orientado hacia la “izquierda” o a la “derecha”, dependiendo de la posición en la que se encuentre el robot.

Cálculo de q_5

De forma geométrica, P_{1y}^6 depende únicamente del ángulo q_5 . Esto se debe a que cuando el sistema de referencia 5 gira un ángulo q_5 sobre su eje z_5 , x_5 forma un ángulo q_5 con x_4 , y debido a la disposición de sistemas colocada anteriormente y_5 tiene la misma dirección y sentido que z_6 , que a su vez forma un ángulo q_5 con z_4 [30]. En la Figura 2.5 se presenta una vista superior del robot manipulador UR5 que permite observar este análisis.

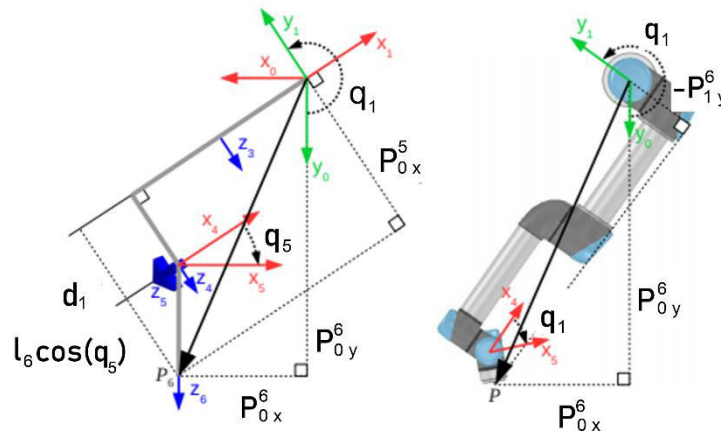


Figura 2.5 Vista superior del robot UR5 y análisis geométrico de P_{1y}^6 , tomado de [30]

Por lo tanto, se puede expresar P_{1y}^6 como:

$$-P_{1y}^6 = D_1 + l_6 \cos q_5 \quad (2.10)$$

Además, P_{1y}^6 puede ser expresada a través de una rotación de P_0^6

$$P_0^6 = R_0^1 P_1^6 \quad (2.11)$$

$$P_1^6 = R_0^{1T} P_0^6 \quad (2.12)$$

$$\begin{bmatrix} P_{1x}^6 \\ P_{1y}^6 \\ P_{1z}^6 \end{bmatrix} = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 \\ \sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} P_{0x}^6 \\ P_{0y}^6 \\ P_{0z}^6 \end{bmatrix} \quad (2.13)$$

$$\begin{bmatrix} P_{1x}^6 \\ P_{1y}^6 \\ P_{1z}^6 \end{bmatrix} = \begin{bmatrix} \cos(q_1) & \sin(q_1) & 0 \\ -\sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{0x}^6 \\ P_{0y}^6 \\ P_{0z}^6 \end{bmatrix} \quad (2.14)$$

$$P_{1y}^6 = P_{0x}^6(-\sin q_1) + P_{0y}^6 \cos q_1 \quad (2.15)$$

Entonces, igualando las ecuaciones planteadas anteriormente, se obtiene:

$$P_{0x}^6(-\sin q_1) + P_{0y}^6(\cos q_1) = -D_1 - l_6 \cos q_5 \quad (2.16)$$

$$q_5 = \pm \arccos \left(\frac{P_{0x}^6 \sin q_1 - P_{0y}^6 \cos q_1 - D_1}{l_6} \right) \quad (2.17)$$

Cálculo de q_6

Se analiza el eje \hat{y}_1^6 que es paralelo a los ejes $\hat{z}_{2,3,4}^6$, tal y como se puede observar en la Figura 2.6:

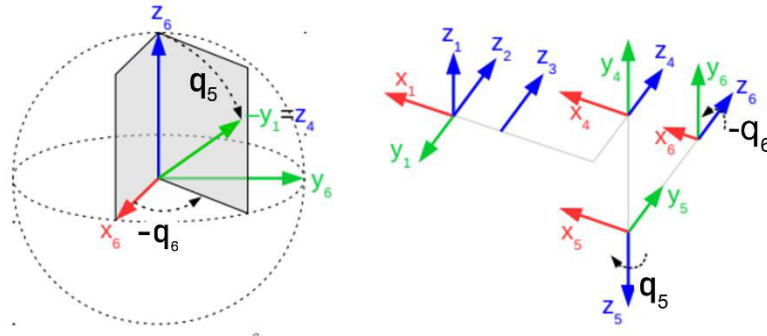


Figura 2.6 Eje $-\hat{y}_1^6$ expresado en coordenadas esféricas, tomado de [30]

El eje \hat{y}_1^6 depende únicamente de q_5 y q_6 , el ángulo azimutal y polar, respectivamente. Entonces, si se transforma $-\hat{y}_1^6$ de coordenadas esféricas a coordenadas cartesianas se tiene:

$$-\hat{y}_1^6 = \begin{bmatrix} \sin(q_5) \cdot \cos(-q_6) \\ \sin(q_5) \cdot \sin(-q_6) \\ \cos(q_5) \end{bmatrix} \quad (2.18)$$

$$\hat{y}_1^6 = \begin{bmatrix} -\sin(q_5) \cdot \cos(q_6) \\ \sin(q_5) \cdot \sin(q_6) \\ -\cos(q_5) \end{bmatrix} \quad (2.19)$$

Además, \hat{y}_1^6 puede ser expresada mediante una rotación de q_1 en el plano x - y del sistema de referencia base de la siguiente manera:

$$\hat{y}_0^6 = R_0^1 \hat{y}_1^6 \quad (2.20)$$

$$\hat{y}_1^6 = R_0^1{}^T \hat{y}_0^6 \quad (2.21)$$

$$\begin{bmatrix} \hat{x}_1^6 \\ \hat{y}_1^6 \\ \hat{z}_1^6 \end{bmatrix} = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 \\ \sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} \hat{x}_0^6 \\ \hat{y}_0^6 \\ \hat{z}_0^6 \end{bmatrix} \quad (2.22)$$

$$\begin{bmatrix} \hat{x}_1^6 \\ \hat{y}_1^6 \\ \hat{z}_1^6 \end{bmatrix} = \begin{bmatrix} \cos(q_1) & \sin(q_1) & 0 \\ -\sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_0^6 \\ \hat{y}_0^6 \\ \hat{z}_0^6 \end{bmatrix} \quad (2.23)$$

$$\hat{y}_1^6 = \hat{x}_0^6(-\sin q_1) + \hat{y}_0^6(\cos q_1) \quad (2.24)$$

$$\hat{y}_1^6 = \begin{bmatrix} -\hat{x}_{0_x}^6 (\sin q_1) + \hat{y}_{0_x}^6 (\cos q_1) \\ -\hat{x}_{0_y}^6 (\sin q_1) + \hat{y}_{0_y}^6 (\cos q_1) \\ -\hat{x}_{0_z}^6 (\sin q_1) + \hat{y}_{0_z}^6 (\cos q_1) \end{bmatrix} \quad (2.25)$$

Igualando las Ecuaciones (2.19) y (2.25), se obtiene:

$$-\sin q_5 \cos q_6 = -\hat{x}_{0_x}^6 (\sin q_1) + \hat{y}_{0_x}^6 (\cos q_1) \quad (2.26)$$

$$\sin q_5 \sin q_6 = -\hat{x}_{0_y}^6 (\sin q_1) + \hat{y}_{0_y}^6 (\cos q_1) \quad (2.27)$$

$$q_6 = \text{atan2}\left(\frac{-\hat{x}_{0_y}^6 (\sin q_1) + \hat{y}_{0_y}^6 (\cos q_1)}{-\sin q_5}, \frac{-\hat{x}_{0_x}^6 (\sin q_1) + \hat{y}_{0_x}^6 (\cos q_1)}{\sin q_5}\right) \quad (2.28)$$

Cálculo de q_3

Finalmente, para hallar los ángulos de las articulaciones restantes, se observa que todas las articulaciones son paralelas una respecto a otra. Por lo tanto, se analizan como si el manipulador robótico UR5 fuera un robot planar de tres grados de libertad, tal y como se muestra en la Figura 2.7.

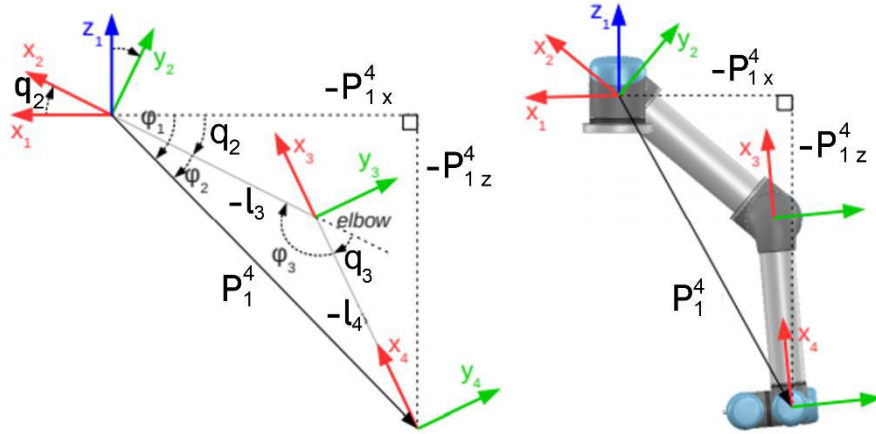


Figura 2.7 Articulaciones 2, 3 y 4 del Manipulador Robótico UR5, tomado de [30]

Para hallar q_3 , se calcula T_1^4 , que se obtiene fácilmente si se conoce T_0^1 , T_4^5 , T_5^6 , por lo que se calcula a través de la siguiente ecuación:

$$T_{i-1}^i = \begin{bmatrix} \cos(q_i) & -\cos(\alpha_i) \sin(q_i) & \sin(\alpha_i) \sin(q_i) & a_i \cos(q_i) \\ \sin(q_i) & \cos(\alpha_i) \cos(q_i) & -\sin(\alpha_i) \cos(q_i) & a_i \sin(q_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.29)$$

Además, se conoce que la matriz de transformación homogénea de un sistema de referencia puede ser expresado como un producto de dos o más matrices de transformación en muchas formas, por ejemplo:

$$T_U^D = T_U^A T_A^D \quad (2.30)$$

$$T_U^D = T_U^B T_B^C T_C^D \quad (2.31)$$

Finalmente, se conoce que:

$$T_A^B = T_B^A^{-1} \quad (2.32)$$

Entonces:

$$T_1^4 = T_1^0 T_0^6 T_6^5 T_5^4 \quad (2.33)$$

$$T_1^4 = \begin{bmatrix} \hat{x}_{1x}^4 & \hat{y}_{1x}^4 & \hat{z}_{1x}^4 & p_x \\ \hat{x}_{1y}^4 & \hat{y}_{1y}^4 & \hat{z}_{1y}^4 & p_y \\ \hat{x}_{1z}^4 & \hat{y}_{1z}^4 & \hat{z}_{1z}^4 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.34)$$

$$\cos(\varphi_3) = \frac{(-l_3)^2 + (-l_4)^2 - |P_{1\ xz}^4|^2}{2(-l_3)(-l_4)} \quad (2.35)$$

A partir de Figura 2.7, se obtiene la siguiente relación entre φ_3 y q_3 :

$$\cos(q_3) = \cos(180 - \varphi_3) = -\cos(\varphi_3) \quad (2.36)$$

$$q_3 = \pm \arccos\left(\frac{|P_{1\ xz}^4|^2 - (-l_3)^2 - (-l_4)^2}{2(l_3)(l_4)}\right) \quad (2.37)$$

Cálculo de q_2

A partir de Figura 2.7, q_2 se obtiene a partir de $\varphi_1 - \varphi_2$:

$$\varphi_1 = \operatorname{atan2}(-P_{1\ z}^4, -P_{1\ x}^4) \quad (2.38)$$

Y a través de ley de senos, se tiene:

$$\frac{\sin(\varphi_2)}{-l_4} = \frac{\sin(\varphi_3)}{|P_{1\ xz}^4|} \quad (2.39)$$

$$q_2 = \varphi_1 - \varphi_2 = \operatorname{atan2}(-P_{1\ z}^4, -P_{1\ x}^4) - \operatorname{asin}\left(\frac{l_4 \sin(q_3)}{|P_{1\ xz}^4|}\right) \quad (2.40)$$

Cálculo de q_4

El ángulo de la articulación q_4 es el ángulo que existe entre \hat{x}_3 y \hat{x}_4 a lo largo de \hat{z}_4 . Por lo tanto, se obtiene a partir de la matriz de transformación T_3^4 y el vector unitario del eje X del sistema de referencia 4 respecto al sistema de referencia 3.

$$T_3^4 = T_3^2 T_2^1 T_1^4 \quad (2.41)$$

$$q_4 = \operatorname{atan2}(\hat{x}_{3\ y}^4, \hat{x}_{3\ x}^4) \quad (2.42)$$

Las expresiones obtenidas describen el modelo cinemático inverso del manipulador robótico UR5 y permiten obtener una o varias soluciones para que el robot alcance una referencia de posición y orientación deseada [9]. Sin embargo, estas expresiones son de naturaleza no lineal, y su solución no siempre es sencilla o posible desde el punto de vista de la estructura del manipulador [11], por lo que, si este es el caso, se pueden utilizar métodos numéricos o iterativos como otras posibles formas de solución [7], tales como los presentados en [14].

2.2.3 MATRIZ JACOBIANA

La Jacobiana geométrica del manipulador UR5 se divide en la obtención de las submatrices de velocidad angular y lineal, cuyo cálculo utiliza su modelo cinemático directo.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = J(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \quad (2.43)$$

A partir de la Ecuación (2.3) del modelo cinemático directo del robot, se deriva parcialmente respecto al vector de posiciones articulares $q = [q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6]^T$, tal y como se indica en la Sección 1.3.1.2.3, por lo que se obtiene lo siguiente:

$$J_v(q) = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \end{bmatrix} \quad (2.44)$$

La matriz Jacobiana de velocidad angular se calcula siguiendo el procedimiento mostrado en la Sección 1.3.1.2.3, donde se utilizan las matrices de transformación homogénea de las Ecuaciones (2.1) y (2.2) utilizadas en la obtención del modelo cinemático directo del robot.

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = R_0^1 \vec{q}_1 + R_0^2 \vec{q}_2 + R_0^3 \vec{q}_3 + R_0^4 \vec{q}_4 + R_0^5 \vec{q}_5 + R_0^6 \vec{q}_6 = J_\omega \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \quad (2.45)$$

$$J_\omega = \left[R_0^1 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + R_0^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + R_0^3 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + R_0^4 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + R_0^5 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + R_0^6 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] \quad (2.46)$$

$$J_\omega = \begin{bmatrix} J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix} \quad (2.47)$$

Por lo que la matriz Jacobiana geométrica del manipulador robótico UR5 es la siguiente:

$$J(q) = \begin{bmatrix} J_v(q) \\ J_\omega(q) \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix} \quad (2.48)$$

En el Anexo B, se presenta desarrollada completamente la matriz Jacobiana geométrica del manipulador robótico UR5.

2.2.3.1 Matriz Jacobiana Analítica

Para ejecutar una tarea específica con el efector final de un robot manipulador, es necesario especificar su posición y orientación en cada instante, lo que resulta bastante simple para su posición, al contrario que con su orientación, que resulta mucho más complejo, puesto que se especifica a partir de una matriz de rotación formada por tres vectores unitarios mutuamente ortogonales que describen su orientación respecto a su sistema de referencia base, por lo que la ortonormalidad del sistema de referencia asociado al efector final del robot debe ser garantizada en cada instante de tiempo [11].

Por lo tanto, para solucionar este problema es necesario adoptar una representación mínima de coordenadas en el espacio operacional del robot manipulador tanto para su posición, como para su orientación, tales como su posición cartesiana (x, y, z) , y coordenadas de orientación dadas por ángulos de Euler (θ, ϕ, ψ) , respectivamente [10]. Entonces, si se elige este tipo de representación, es necesario el uso de la matriz Jacobiana analítica, la cual se obtiene a partir de diferenciación directa de su modelo cinemático directo, tal y como se muestra en la Ecuación (1.20). Sin embargo, realizar el cálculo de esta matriz a partir de diferenciación no es sencillo, puesto que las funciones que representan los ángulos Euler, dados por el vector $\phi_e = [\theta \ \phi \ \psi]^T$, no están disponibles directamente, sino que es necesario realizar su cálculo a partir de los elementos de la matriz de rotación obtenida del modelo cinemático directo del robot [11]. Debido a esto, se puede encontrar una matriz de transformación $A(\phi_e)$ que permite relacionar la velocidad angular ω de efector final respecto a su sistema de referencia base con la derivada temporal de los ángulos de Euler escogidos para expresar su orientación, y se expresa de la siguiente forma [11]:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = A(\phi_e) \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \quad (2.49)$$

Una vez que la matriz $A(\phi_e)$ sea calculada, la matriz Jacobiana analítica $J_a(q)$ se obtiene a partir de la siguiente ecuación [11]:

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & A(\phi_e)^{-1} \end{bmatrix} J(q) \quad (2.50)$$

Donde I es la matriz identidad de dimensiones 3×3 , y O es una matriz nula de 3×3 , con el fin de no afectar a las coordenadas de velocidad lineal que son las mismas tanto para la Jacobiana geométrica, como para la analítica. Finalmente, $A(\phi_e)^{-1}$ es la inversa de la matriz de transformación $A(\phi_e)$, y $J(q)$ es la matriz Jacobiana geométrica cuyo cálculo se obtiene en la Sección 2.2.3 y en el Anexo B.

La matriz de transformación $A(\phi_e)$ se calcula a partir de la configuración de los ángulos de Euler que se elijan para representar la orientación del manipulador, que en este caso, es la convención XYZ, que implica un desplazamiento angular θ , ϕ , y ψ , en los ejes x , y y z , respectivamente, partiendo de la orientación del sistema de referencia base del robot [11].

Si la orientación del sistema de referencia asociado al efector final de un manipulador parte de la orientación de su sistema de referencia base, esta viene dada por una matriz identidad, dada por la siguiente expresión:

$$R_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.51)$$

Si este sistema de referencia gira alrededor de su eje x con una velocidad $\dot{\theta}$, la contribución a su velocidad angular respecto al sistema de referencia base del robot, está dada por la siguiente expresión:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \dot{\theta} \quad (2.52)$$

Posteriormente, este sistema de referencia que ya ha realizado un desplazamiento angular θ en su eje x , se representa a partir de la siguiente matriz de rotación:

$$R_1 = R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (2.53)$$

Si este nuevo sistema de referencia gira alrededor de su eje y con una velocidad $\dot{\phi}$, la contribución a su misma velocidad angular respecto al sistema de referencia base del robot, viene dada por la siguiente expresión:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix} \dot{\phi} \quad (2.54)$$

El sistema de referencia resultante ha realizado un desplazamiento angular ϕ en su eje y , y se representa a partir de la siguiente matriz de rotación:

$$R_2 = R_x(\theta)R_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ \sin(\theta)\cos(\phi) & \cos(\theta) & -\sin(\theta)\cos(\phi) \\ -\cos(\theta)\cos(\phi) & \sin(\theta) & \cos(\theta)\cos(\phi) \end{bmatrix} \quad (2.55)$$

Si este sistema de referencia resultante realiza un giro alrededor de su eje z con una velocidad $\dot{\psi}$, la contribución a su velocidad angular respecto al sistema de referencia base del robot, viene dada por la siguiente expresión:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \sin(\phi) \\ -\sin(\theta)\cos(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \dot{\psi} \quad (2.56)$$

Por lo tanto, reuniendo cada una de las contribuciones de las velocidades rotacionales de los ángulos de Euler $\dot{\theta}$, $\dot{\phi}$ y $\dot{\psi}$, a la velocidad angular del efector final respecto a su sistema de referencia base, se tiene que la matriz de transformación $A(\phi_e)$ está dada por la siguiente expresión [1]:

$$A(\phi_e) = \begin{bmatrix} 1 & 0 & \sin(\phi) \\ 0 & \cos(\theta) & -\sin(\theta)\cos(\phi) \\ 0 & \sin(\theta) & \cos(\theta)\cos(\phi) \end{bmatrix} \quad (2.57)$$

Por lo que entonces se obtiene:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & \sin(\phi) \\ 0 & \cos(\theta) & -\sin(\theta)\cos(\phi) \\ 0 & \sin(\theta) & \cos(\theta)\cos(\phi) \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \quad (2.58)$$

2.3 DISEÑO DEL ENTORNO VIRTUAL EN COPPELIASIM

La plataforma virtual consiste en la creación de un entorno virtual adecuado para el manipulador robótico UR5, y el desarrollo de tareas de pintura, y así cumpla con sus objetivos de control. En primera instancia, se crea una nueva escena dentro del software de simulación CoppeliaSim. Posteriormente, en la pestaña lateral *Model Browser* de la interfaz de CoppeliaSim, se ingresa en la carpeta *robots* y luego a la subcarpeta *non-mobile*, y se elige el manipulador robótico UR5. Una vez seleccionado, se lo arrastra hacia la escena de CoppeliaSim, tal y como se muestra en la Figura 2.8.

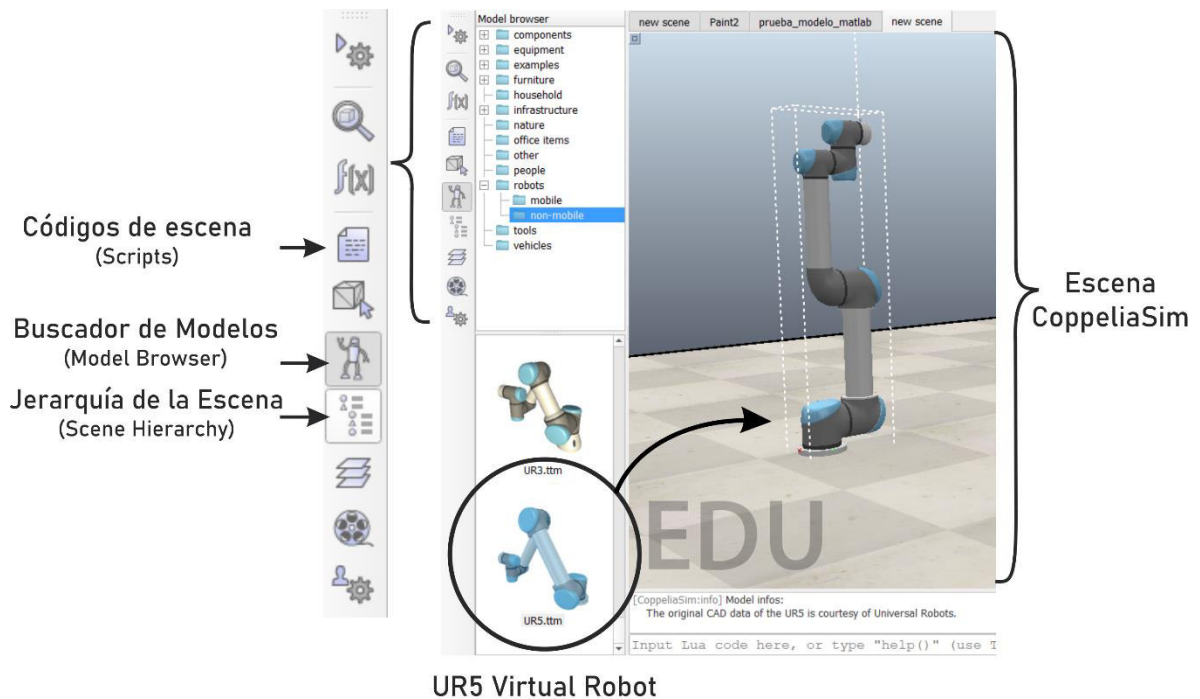


Figura 2.8 Diseño de la escena en CoppeliaSim

Después de esto, se colocan las paredes, y se agregan los elementos necesarios que forman parte del entorno utilizando tanto las herramientas, como los modelos adicionales que dispone por defecto CoppeliaSim.

La Figura 2.9 muestra el entorno virtual desarrollado en CoppeliaSim enfocado en el adecuado funcionamiento del manipulador robótico UR5, para que sea capaz de pintar tanto paredes, como superficies con diferentes disposiciones dentro del entorno, y de esta forma, demostrar el funcionamiento de los controladores diseñados a fin de comparar su desempeño frente a los diferentes perfiles de referencia indicados en la Sección 2.1.

Con el fin de presentar los resultados de las tareas de pintura, se acopló una pistola de pintura o *paint gun* al efector final del modelo virtual del Robot, ver Figura 2.9b, por lo que tanto su posición, como orientación, coinciden con el sistema de referencia correspondiente al efector final del robot.

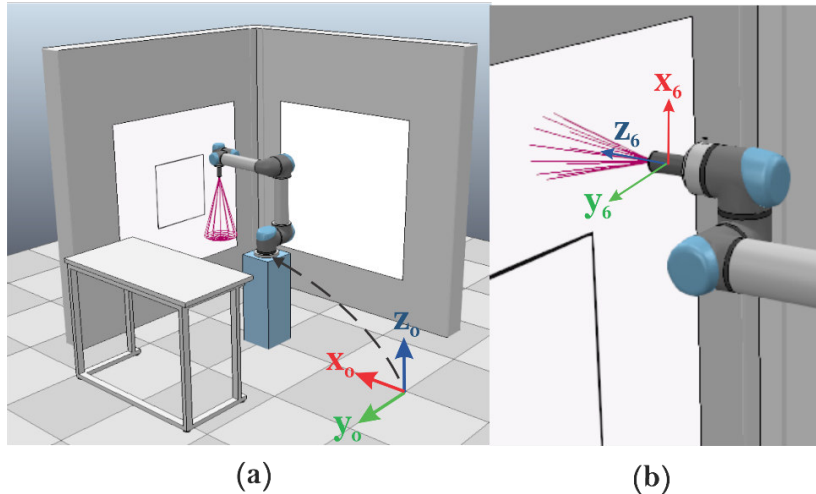


Figura 2.9 Entorno virtual desarrollado en CoppeliaSim a) Manipulador robótico UR5 en posición de reposo b) Pistola de pintura acoplada al efector final del UR5

2.4 IMPLEMENTACIÓN DE LOS ALGORITMOS DE CONTROL

El robot virtual UR5 en CoppeliaSim está controlado por las técnicas de control propuestas en la Sección 1.3.3, que se implementan en Matlab, por lo que, la comunicación entre ambas aplicaciones se realiza a través de una Interfaz de Programación de Aplicaciones (API), mostrada en la Sección 1.3.4.1. En la Figura 2.10, se muestra el esquema general del sistema de control utilizado.

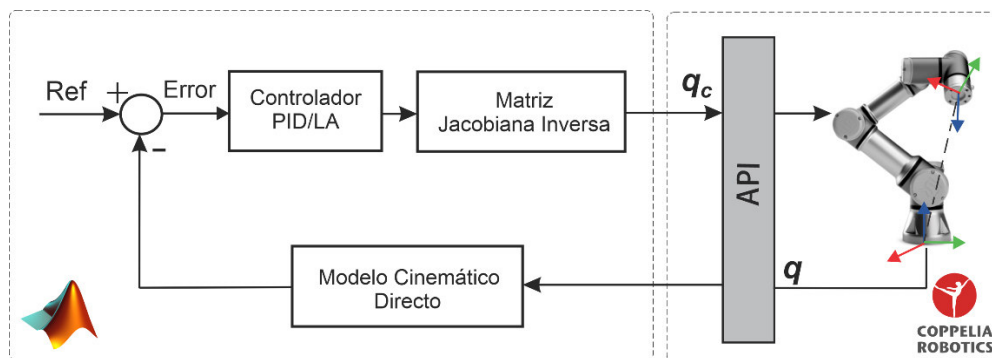


Figura 2.10 Diagrama de bloques del sistema de control del Manipulador Robótico Virtual UR5

El sistema de control mostrado en la Figura 2.10 se divide en dos bloques principales, el bloque implementado en Matlab, y el de CoppeliaSim. En Matlab, se implementa el modelo cinemático directo del robot, el cual se utiliza para obtener su estado actual, esto incluye posición y orientación de su efector final, y posteriormente, para obtener su error de posición y orientación dependiendo del perfil de referencia que se utilice como entrada del sistema de control. Adicionalmente, también se implementa la inversa de la matriz

Jacobiana analítica, obtenida en la Sección 2.2.3, ya que forma una parte fundamental de las técnicas de control propuestas. Por otra parte, en CoppeliaSim, se tiene tanto la interfaz de comunicación, como el entorno virtual desarrollado en la Sección 2.3.

Las acciones de control calculadas, descritas por el vector de velocidades articulares $q_c = [q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6]$, se envían hacia el modelo virtual del robot UR5 en CoppeliaSim para que alcance con sus objetivos de control. En Matlab, se utiliza el comando *simxSetJointTargetVelocity* que permite fijar una velocidad angular en radianes por segundo en cualquier articulación rotacional presente en el entorno virtual de CoppeliaSim [18]. Y se realiza a partir de la siguiente línea de código:

$$vrep.simxSetJointTargetVelocity(ID, Tag, Velocidad, Modo);$$

Donde *ID* es la dirección del cliente dentro de la interfaz de comunicación, *Tag* es la etiqueta con la que se declaró una articulación rotacional, *Velocidad* es el valor numérico en rad/s que se fijarán en la articulación, y *Modo* indica el formato de comunicación que tendrá este comando, que puede ser sincrónica o asincrónica [17].

La matriz Jacobiana inversa, y el estado actual del robot, dependen principalmente de las posiciones articulares del robot, descritas por el vector de posiciones articulares $q = [q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6]$, que se realimenta hacia Matlab a través de la interfaz de comunicación antes mencionada. Por lo tanto, en Matlab, se utiliza el comando *simxGetJointPosition* que permite leer la posición angular de articulaciones rotacionales presentes en el entorno virtual [18] y se realiza a partir de la siguiente línea de código:

$$[Estado, Valor] = vrep.simxGetJointPosition(ID, Tag, Modo);$$

Donde *Estado* indica el estado de la comunicación que adopta un código numérico específico en caso de presentarse un error, y *Valor* es una variable auxiliar que almacena la posición angular de la articulación apuntada una vez que el comando haya sido ejecutado con éxito [17]. En la Figura 2.11, se muestran los comandos de comunicación utilizados para interactuar con el robot virtual en CoppeliaSim desde Matlab.

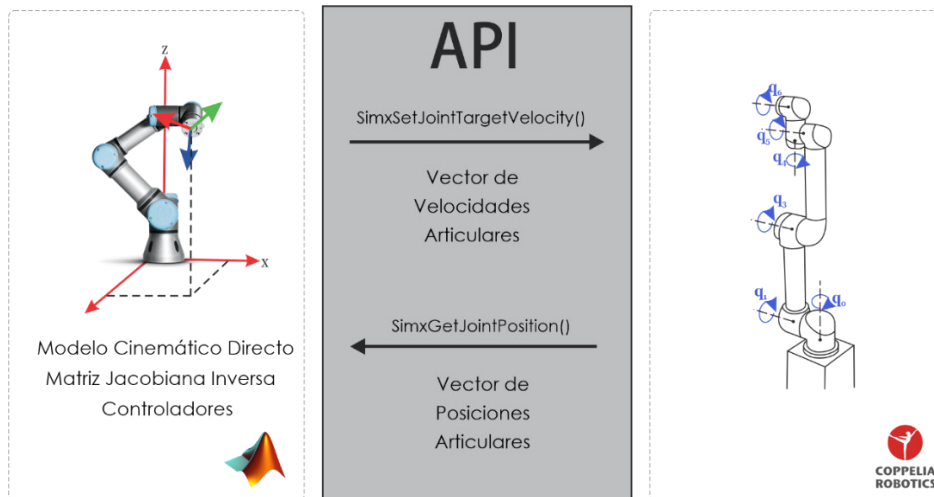


Figura 2.11 Comandos de comunicación entre Matlab y CoppeliaSim utilizados en el sistema de control

Finalmente, se muestra en la Figura 2.12 el diagrama de flujo del sistema de control propuesto durante la simulación del robot en CoppeliaSim.

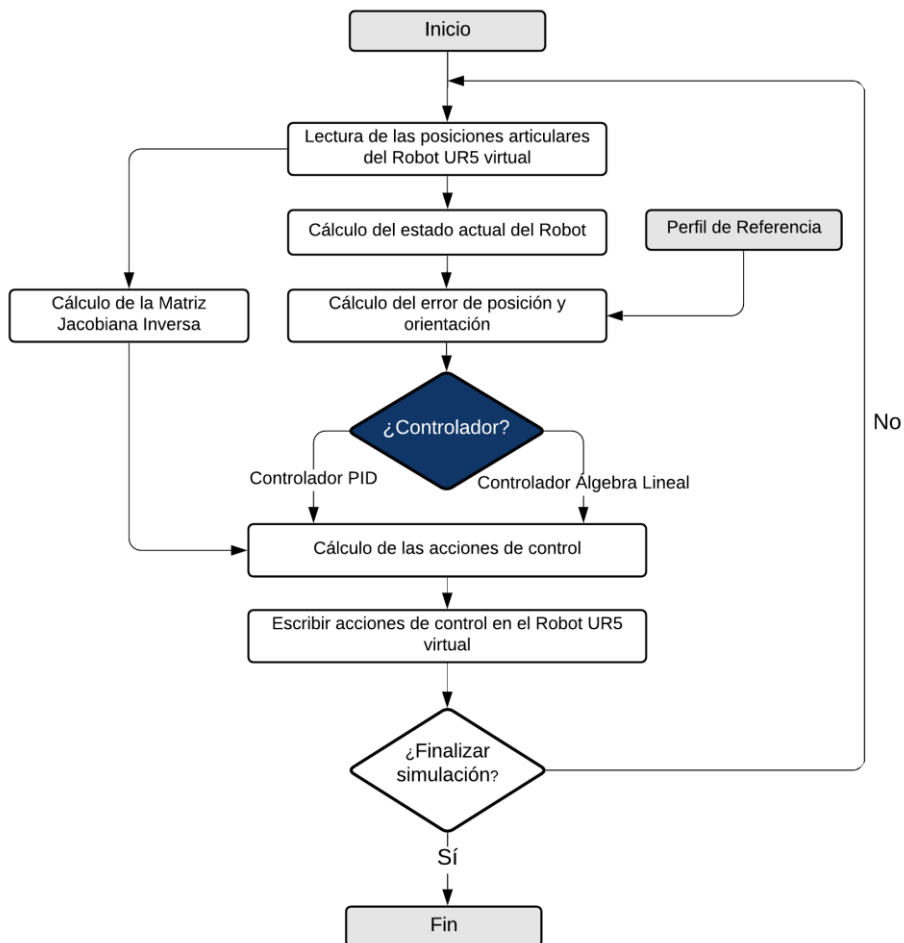


Figura 2.12 Diagrama de flujo del funcionamiento del sistema de control

En el Anexo C, se muestra a detalle la configuración necesaria para establecer la comunicación entre Matlab y CoppeliaSim a través de una API.

2.5 INTEGRACIÓN DEL SISTEMA DE RECONOCIMIENTO

El sistema de reconocimiento de gestos utilizado en este proyecto es el modelo general del sistema desarrollado en el Proyecto de Investigación PIGR-19-07, que permite identificar 5 gestos diferentes de la mano, con un rendimiento de hasta el 81,3 % [5].

Adicionalmente, tanto el sistema de reconocimiento, como los modelos matemáticos del robot y sus controladores, se encuentran implementados en Matlab. Entonces, esto dificultaría que ambos procesos puedan ejecutarse en una sola instancia de Matlab debido a la complejidad de sus tareas y cálculos. Por lo tanto, en la Figura 2.13 se presenta el esquema de comunicación del sistema de reconocimiento de gestos con el bloque de Matlab del sistema de control, presentado en la Sección 2.4, que funcionan en instancias independientes de Matlab.

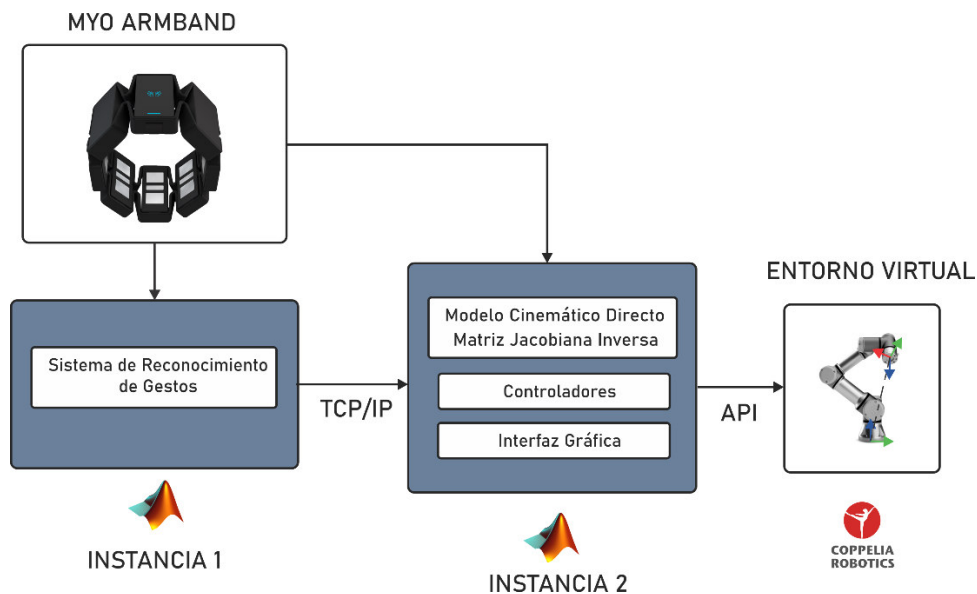


Figura 2.13 Esquema de comunicación entre el sistema de reconocimiento de gestos y el sistema de control del robot manipulador virtual UR5

El protocolo de comunicación utilizado entre las instancias de Matlab es TCP/IP que se configura de la siguiente forma:

- El servidor encargado de proveer de información al cliente es la Instancia 1, que ejecuta el sistema de reconocimiento de gestos.
- La Instancia 2 es el cliente que recibe como dato el gesto clasificado por parte del servidor.

- Adicionalmente, el sensor Myo Armband se conecta a ambas instancias, y envía las señales EMG y ángulos de Euler de su IMU a la Instancia 1 e Instancia 2, respectivamente.

A continuación, se detallan las funciones asignadas de cada uno de los gestos clasificados por el sistema de reconocimiento de gestos.

2.5.1 GESTOS Y COMANDOS DE SELECCIÓN

Para alcanzar los objetivos de control y cumplir con la aplicación de pintura presentada en la Sección 2.3, se elige el gesto FIST para activar o desactivar la pistola de pintura acoplada al efector final del robot UR5. Adicionalmente, el robot debe alcanzar diferentes ubicaciones con orientaciones específicas dentro de su espacio de trabajo, por lo que se utilizan los gestos WAVEIN y WAVEOUT como comandos de selección de las tres diferentes referencias de orientación presentadas en la Figura 2.14, que van en concordancia con la ubicación de las superficies establecidas en el entorno virtual diseñado en la Sección 2.3. Estas referencias de orientación, servirán como entrada del sistema de control presentado en la Sección 2.4.

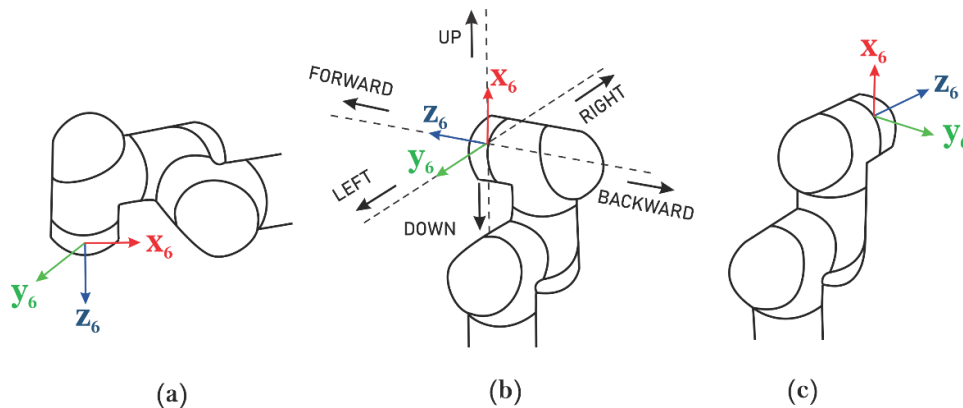


Figura 2.14 Movimiento y referencias de orientación del efector final del robot UR5 a) dirección en el eje z del sistema de referencia base b) dirección en el eje x del sistema de referencia base c) dirección en el eje y del sistema de referencia base

La referencia de posición del efector final del robot dentro de su espacio de trabajo se establece a través de comandos de movimiento generados por los ángulos de Euler proporcionados por el sensor Myo Armband. Sin embargo, para evitar cambios simultáneos de referencias de posición y orientación en el sistema de control, se establece el gesto OPEN que permite seleccionar entre cambios de referencia de posición o de orientación.

A continuación, se presentan los comandos de movimiento generados a partir de los ángulos de Euler proporcionados por la IMU del sensor Myo Armband.

2.5.2 COMANDOS DE MOVIMIENTO

Los comandos de movimiento cambian la referencia de posición del efector final del manipulador robótico virtual UR5, los cuales se generan a partir de establecer un límite superior e inferior a cada uno de los ángulos de Euler que nos proporciona el sensor Myo Armband. El ángulo pitch permite trasladar el efector final a lo largo del eje z del sistema de referencia base del robot, mientras que el ángulo yaw y roll, permiten cambiar la referencia en sus ejes x y y , respectivamente. Básicamente, si un ángulo alcanza su límite superior, la referencia de posición cambiará provocando que el efector final se mueva en una dirección, pero si alcanza su límite inferior, éste se moverá en la dirección contraria. Si no se supera ningún límite, el efector final permanecerá inmóvil. Los límites y comandos de movimiento para cada ángulo de Euler se presentan la Tabla 2.3 y en la Figura 2.15.

Tabla 2.3 Comandos de Movimiento generados a partir de la IMU del Myo Armband

Ángulos de Euler	Inferior	Superior	Eje	Límites
Pitch	DOWN	UP	Z	$\pm 45^\circ$
Yaw	LEFT	RIGHT	Y	$\pm 45^\circ$
Roll	BACKWARD	FORWARD	X	$\pm 60^\circ$

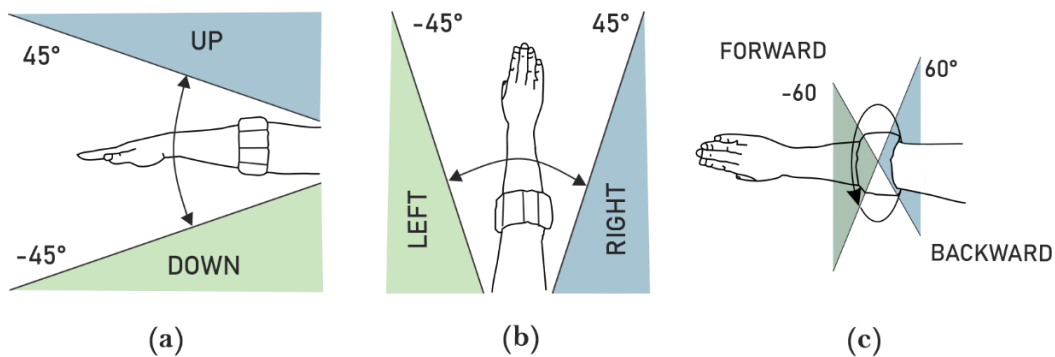


Figura 2.15 Comandos de movimiento generados por la IMU del sensor Myo Armband a) UP y DOWN b) LEFT y RIGHT c) FORWARD y BACKWARD

Finalmente, cabe aclarar que el movimiento del efector final a lo largo de sus ejes es uniforme y tiene una velocidad lineal constante de 0.1 m/s. Si se ejecuta el gesto PINCH, su velocidad de referencia aumentará a 0.2 m/s, la cual volverá a 0.1 m/s si se realiza el gesto nuevamente.

2.6 DESARROLLO DE LA INTERFAZ GRÁFICA

La interfaz gráfica que permite al usuario interactuar con el manipulador robótico virtual se desarrolló en App Designer de Matlab. Su pantalla inicial, mostrada en la Figura 2.16, posee dos paneles principales, el de control y el de monitoreo.

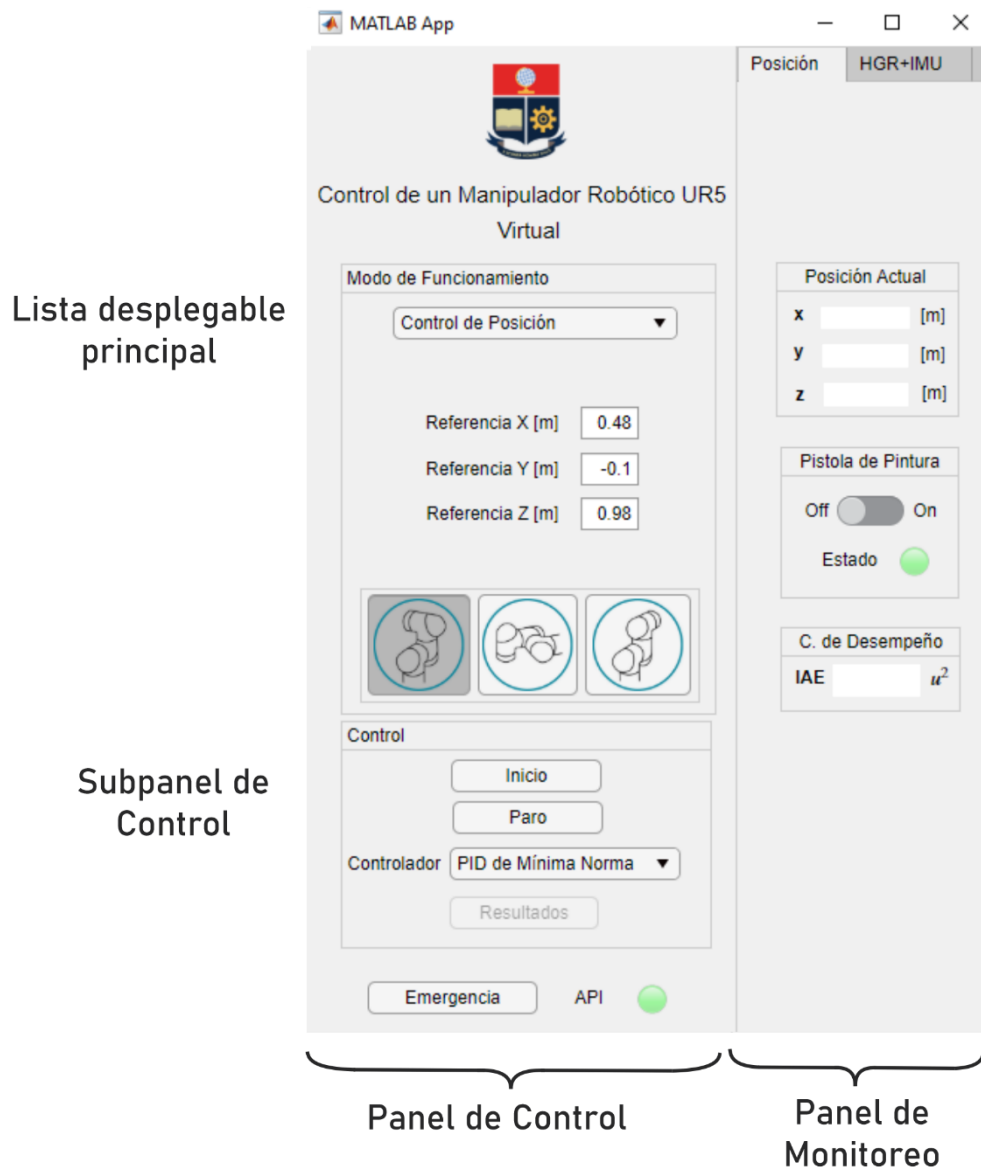


Figura 2.16 Interfaz gráfica desarrollada en App Designer de Matlab y sus paneles principales

El panel de control tiene una lista desplegable de opciones que permite elegir entre tres diferentes modos de funcionamiento para manejar el robot manipulador, ver Figura 2.17, control de posición y orientación, seguimiento de trayectorias, y el modo HGR+IMU. Adicionalmente, posee un sub panel de control que permite comenzar o finalizar la simulación, y seleccionar el controlador a utilizar durante la misma.

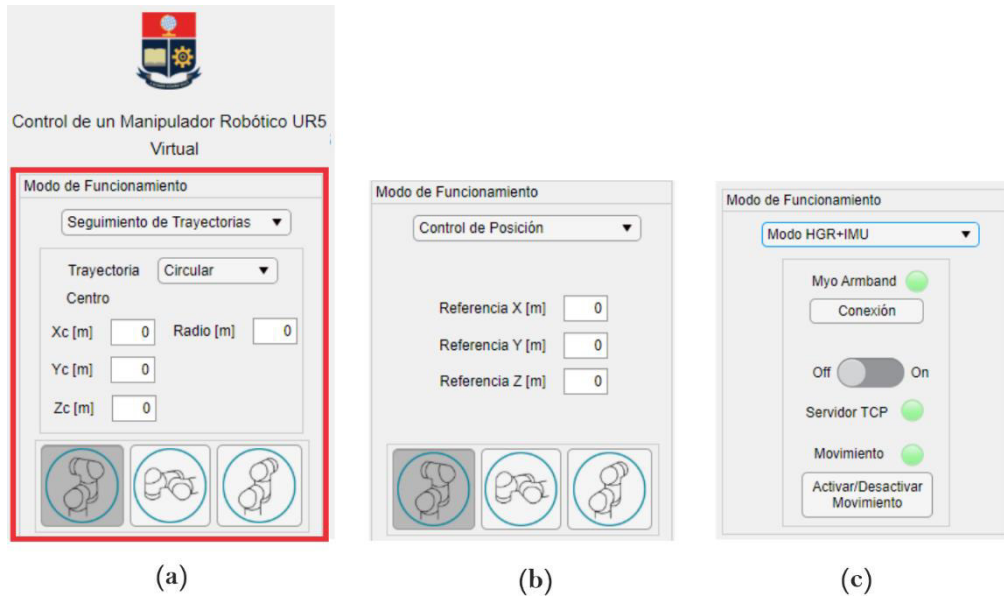


Figura 2.17 Sub panel de Modo de Funcionamiento para a) Seguimiento de Trayectorias b) Control de Posición y Orientación c) Modo HGR+IMU

El panel de monitoreo se utiliza para visualizar el estado actual del efector final del robot, y sus indicadores dependen del modo de funcionamiento elegido. Para el control de posición y seguimiento de trayectorias se presenta la posición actual del efector final y el estado de la pistola de pintura, pero en el modo HGR+IMU, presenta indicadores tanto de gestos como de comandos de movimiento ejecutados, ver Figura 2.18. Adicionalmente, también se indica la referencia de orientación actual del robot y su posición actual.

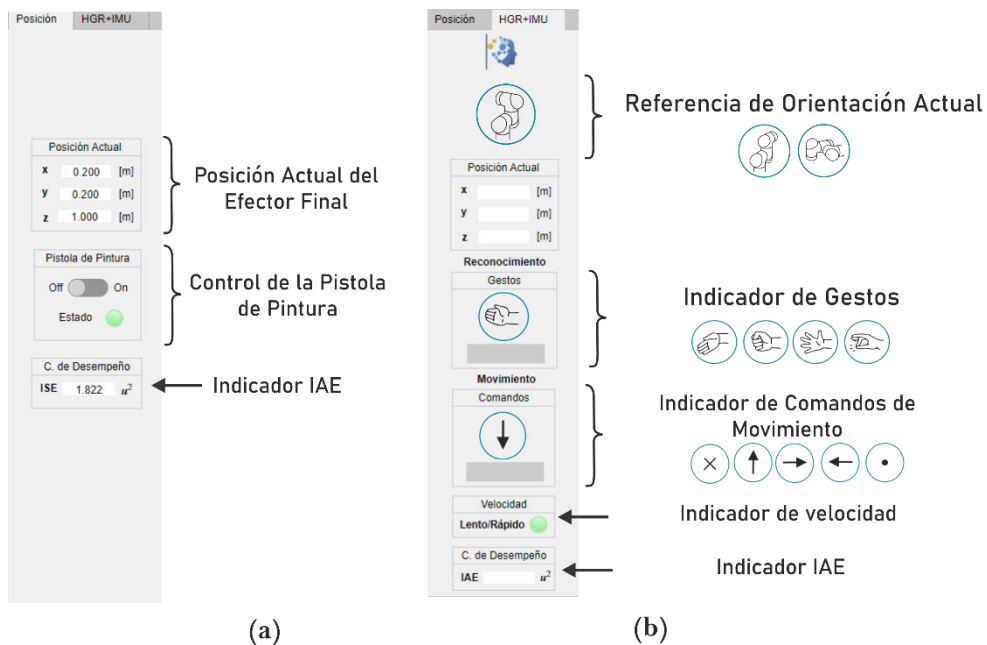


Figura 2.18 Panel de monitoreo de acuerdo al modo de funcionamiento seleccionado a) Control de Posición y Orientación, y Seguimiento de Trayectorias b) Modo HGR+IMU

En todos los modos de funcionamiento, luego de finalizar una simulación, se presenta un indicador numérico que muestra el índice de desempeño IAE calculado durante la misma, a fin de realizar pruebas y comparaciones de rendimiento posteriormente, ver Figura 2.18.

La interfaz gráfica posee un botón de “resultados” que se activa una vez que finaliza una simulación, sin importar el modo de funcionamiento seleccionado, ver Figura 2.19. Al presionarlo, se abre una ventana emergente mostrando los resultados de la simulación, es decir, respuestas del sistema, referencias, y errores en los ejes x , y y z de forma gráfica.

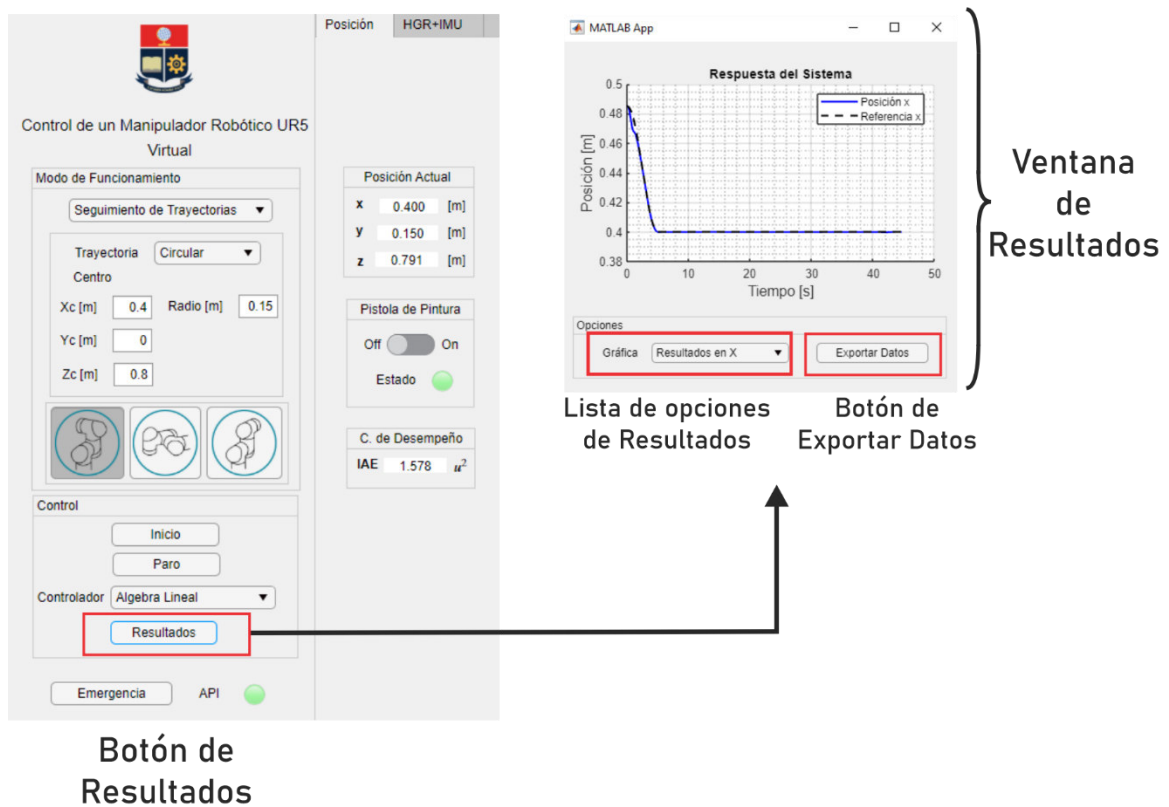


Figura 2.19 Botón y ventana de presentación de resultados de la interfaz gráfica

Adicionalmente, la ventana de resultados posee un botón “Exportar Datos”, ver Figura 2.19, que permite generar un archivo .mat con los resultados de la simulación.

A continuación, se detalla cada uno de los modos de funcionamiento del robot manipulador:

Control de Posición y Orientación

El control de posición consiste en ubicar el efector final del manipulador robótico UR5 en un punto de referencia dado, con una orientación deseada. Básicamente, el objetivo de control consiste en ubicar el efector final del robot de una posición inicial en el punto 1, hasta la posición deseada en el punto 2 [31].

Seguimiento de Trayectorias

El seguimiento de trayectorias consiste en conseguir que el efector final del manipulador robótico virtual UR5 siga como referencia una curva parametrizada en el tiempo a través de una ley de control que permita obtener error cero durante los estados variantes con el tiempo de la trayectoria a seguir [31].

Las trayectorias utilizadas para cumplir con el objetivo de este modo de funcionamiento se forman a partir de un conjunto de coordenadas del espacio operacional del robot, tales como su posición cartesiana (x, y, z) , y coordenadas de orientación dadas por su matriz de rotación o ángulos de Euler (θ, ϕ, ψ) , generados a partir de las funciones básicas disponibles en Matlab.

Modo HGR+IMU

El modo HGR+IMU consiste en conseguir que el efector final del manipulador robótico virtual UR5 siga como referencia un conjunto de coordenadas de posición y orientación obtenidos a partir de los comandos de selección y movimiento indicados en la Sección 2.5, que se generan a partir de los gestos clasificados por el sistema de reconocimiento de gestos, y los ángulos de Euler proporcionados por la IMU del sensor Myo Armband.

El diagrama de flujo de la interfaz gráfica desarrollada se incluye en el Anexo D. De igual manera, en el Anexo E se adjunta el manual de usuario de como inicializar la interfaz, y los pasos a seguir para poner en marcha correctamente el sistema.

3 RESULTADOS Y DISCUSIÓN

En este capítulo se realiza, en primer lugar, la validación del modelo cinemático directo y matriz Jacobiana del manipulador robótico UR5, obtenidos en la Sección 2.2.1 y 2.2.3, respectivamente, con su modelo virtual en CoppeliaSim. Posteriormente, se presenta el procedimiento realizado para encontrar los parámetros de calibración de los controladores propuestos, y finalmente, se detallan las pruebas realizadas con el sistema implementado en la Sección 2.1, donde se presentan sus resultados a través de los diferentes modos de funcionamiento del robot presentados en la Sección 2.6.

Los controladores implementados son:

- Control PID basado en mínima norma
- Control basado en álgebra lineal (LAC)

Las pruebas realizadas con cada uno de los controladores diseñados pueden dividirse principalmente en los siguientes grupos:

- Sin el sistema de reconocimiento de gestos
- Con el sistema de reconocimiento de gestos

Las pruebas que se realizan sin el sistema de reconocimiento de gestos consisten en la utilización de la interfaz gráfica en sus modos de control de posición y orientación, y seguimiento de trayectorias para generar las referencias de posición y orientación del robot UR5 virtual, ver Sección 2.6. Por otra parte, las pruebas con el sistema de reconocimiento de gestos se realizan con el modo HGR+IMU en conjunto con el sensor Myo Armband, tal y como se indica en la Sección 2.5. En cada una de las pruebas se presentan los resultados de la simulación, y se evalúa el funcionamiento del sistema, especialmente en cuanto a los sistemas de control diseñados a través del análisis de los errores de posición, velocidad y el cálculo del Error Absoluto Integrado (IAE).

3.1 VALIDACIÓN DEL MODELO CINEMÁTICO

Para validar el modelo cinemático del robot, que está dado por su matriz Jacobiana presentada en la Sección 2.2.3, se varían los parámetros de entrada del sistema, dados por su vector de velocidades articulares $q_c = [\dot{q}_1 \quad \dot{q}_2 \quad \dot{q}_3 \quad \dot{q}_4 \quad \dot{q}_5 \quad \dot{q}_6]^T$. Esto se debe realizar tanto al manipulador robótico virtual UR5 en CoppeliaSim, como al modelo cinemático del mismo implementado en Matlab a fin de observar si las salidas del sistema

dadas por el vector $[\dot{x} \ \dot{y} \ \dot{z} \ \omega_x \ \omega_y \ \omega_z]^T$ del efector final en cada caso, son similares. El esquema de la prueba de validación se muestra en la Figura 3.1.

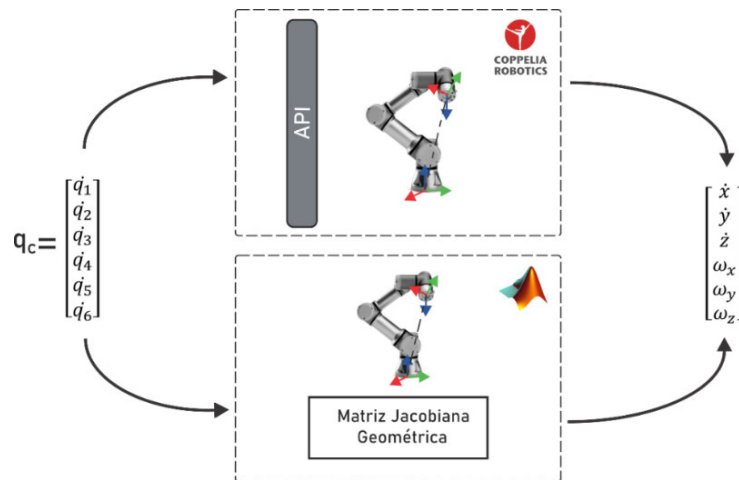


Figura 3.1 Esquema de validación del modelo cinemático del manipulador robótico UR5 con su modelo virtual de CoppeliaSim

En primer lugar, se valida el modelo cinemático directo desarrollado en la Sección 2.2.1 al ingresar una velocidad constante de 0.5 rad/s y 1 rad/s en el articulación 1 y 5, respectivamente, para un tiempo de simulación de 20 segundos, mientras que las demás articulaciones tienen una velocidad nula, es decir $[q_1(t) \ q_2(t) \ q_3(t) \ q_4(t) \ q_5(t) \ q_6(t)]^T = [0.5 \ 0 \ 0 \ 0 \ 1 \ 0]^T$. Adicionalmente, cabe señalar que la posición inicial del efector final del robot es $x_0 = 0.4850 \text{ m}$, $y_0 = -0.10 \text{ m}$, $z_0 = 0.98 \text{ m}$, dada por su posición de reposo, ver Figura 2.9a.

Los resultados de esta prueba, dan como resultado la trayectoria mostrada en la Figura 3.2, tanto a partir del modelo cinemático directo como en CoppeliaSim.

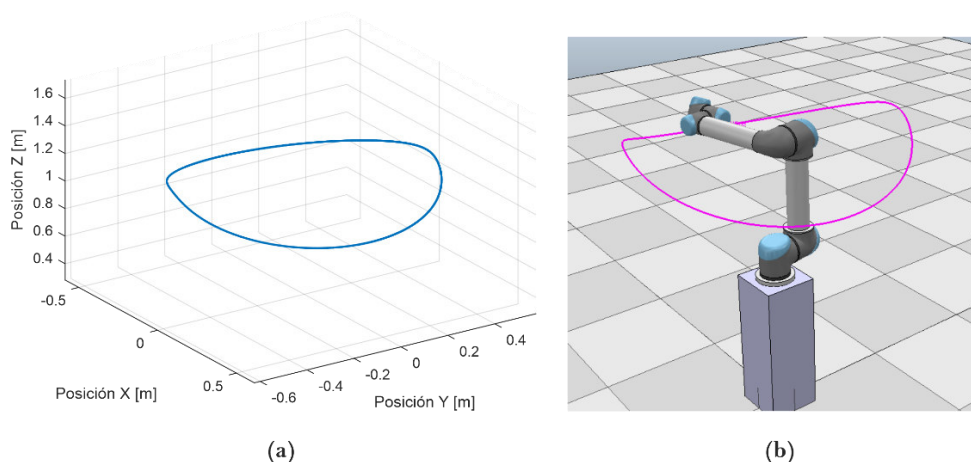


Figura 3.2 Resultados de la validación a) Trayectoria obtenida a partir del modelo cinemático directo b) Trayectoria obtenida del modelo virtual en CoppeliaSim

La respuesta del robot UR5, tanto de su modelo cinemático directo, como de su modelo virtual para esta prueba poseen el mismo comportamiento, ver Figura 3.2, siendo muy similares entre sí, por lo que los resultados presentados en las Figura 3.3 con las respuestas del sistema en sus ejes x , y y z , son favorables. Adicionalmente, también se comparan ambos resultados directamente, en donde el error obtenido presenta un comportamiento prácticamente nulo en todas sus coordenadas, ver Figura 3.4, por lo que el resultado de esta validación es satisfactorio y por ende el modelo cinemático directo, obtenido en la Sección 2.2.1, puede ser utilizado sin ningún inconveniente dentro del sistema de control propuesto.

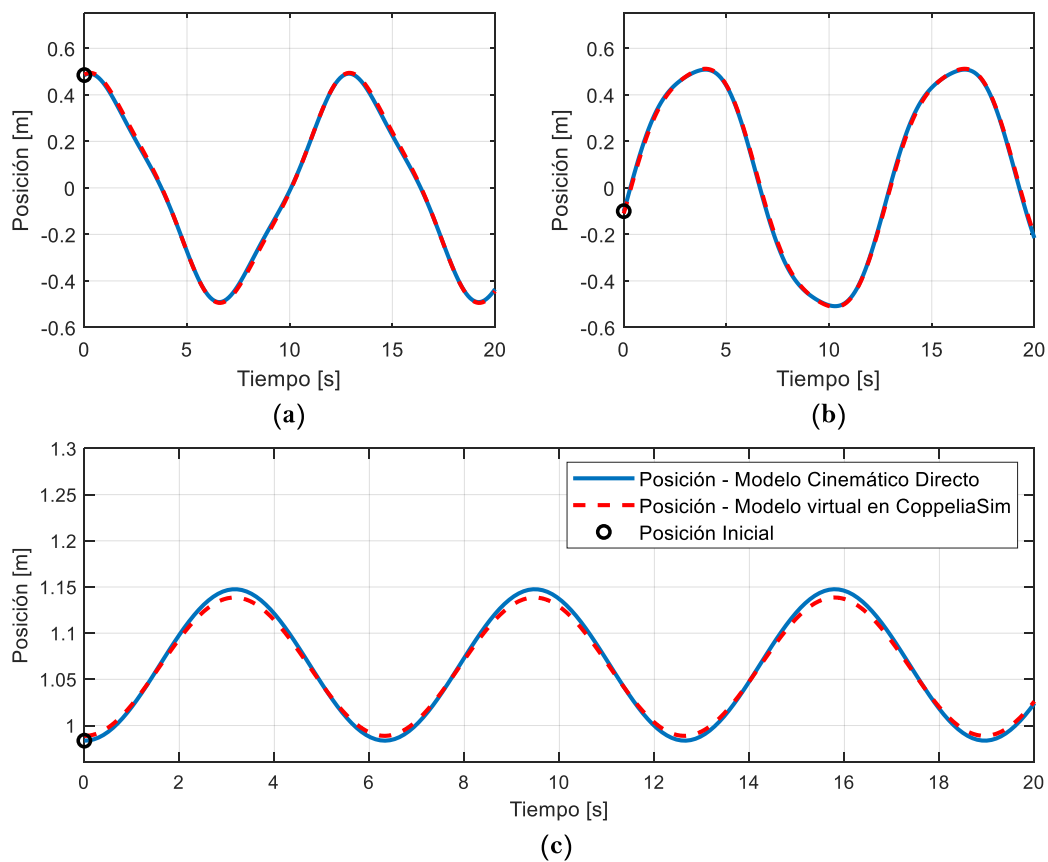


Figura 3.3 Respuesta del modelo cinemático directo del Manipulador Robótico UR5 y de su modelo virtual en CoppeliaSim a) Posición en el eje X b) Posición en el eje Y c) Posición en el eje Z

Además, también se comparan los resultados obtenidos a partir de la matriz Jacobiana del robot y su modelo virtual, es decir, velocidad lineal y angular del efector final del robot, en donde se presenta el error resultado de esta comparación en la Figura 3.5 y Figura 3.6, respectivamente.

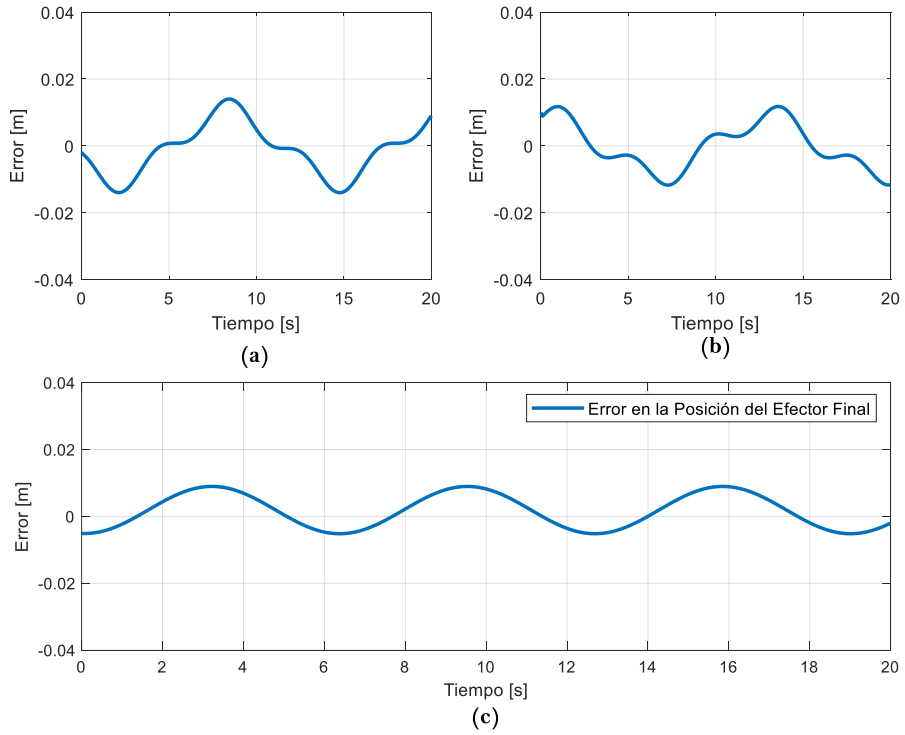


Figura 3.4 Error entre el modelo cinemático directo del Manipulador Robótico UR5 y su modelo virtual en CoppeliaSim a) Error en el eje X b) Error en el eje Y c) Error en el eje Z

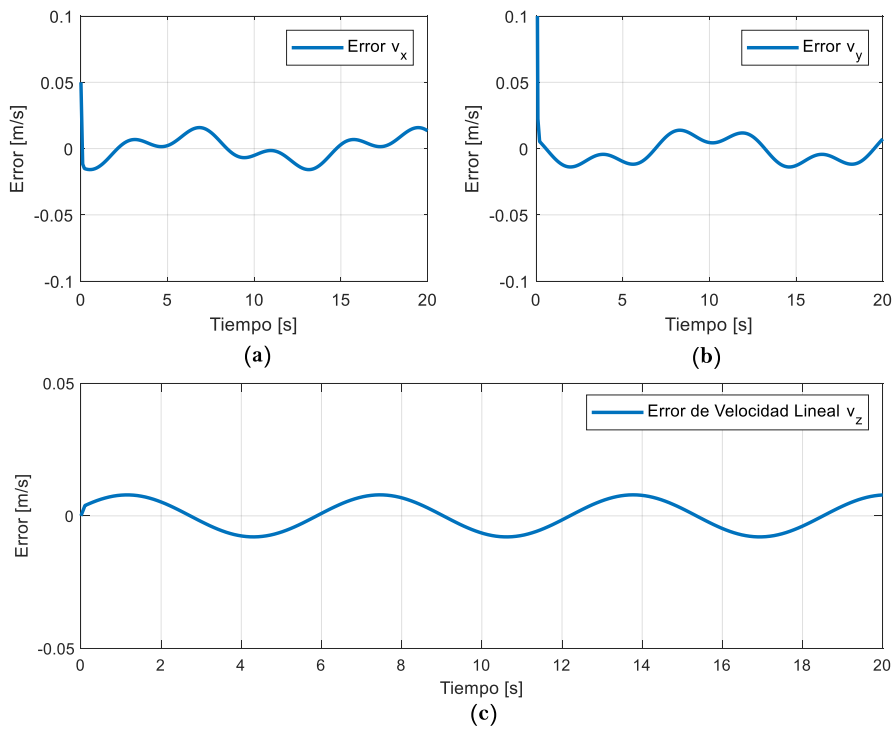


Figura 3.5 Error entre la matriz Jacobiana lineal y su modelo virtual de CoppeliaSim a) Velocidad lineal en el eje X b) Velocidad lineal en el eje Y c) Velocidad lineal en el eje Z

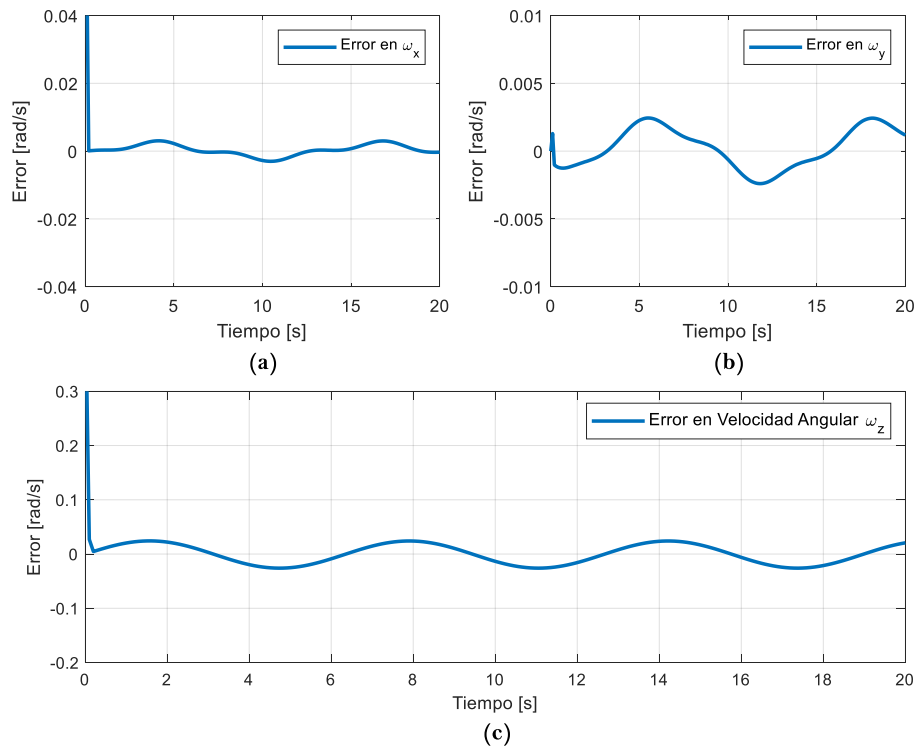


Figura 3.6 Error entre la matriz Jacobiana angular del Manipulador Robótico UR5 y su modelo virtual de CoppeliaSim a) Velocidad alrededor del eje X b) Velocidad alrededor del eje Y c) Velocidad alrededor del eje Z

Los errores obtenidos a partir de la comparación de velocidad lineal y angular obtenidas a partir de la matriz Jacobiana calculada en la Sección 2.2.3, presentan el comportamiento esperado frente al modelo virtual, es decir, son prácticamente nulos, ver Figura 3.5 y Figura 3.6, por lo que el resultado de esta validación, también es satisfactorio.

3.2 CALIBRACIÓN DE LOS CONTROLADORES

Las constantes de calibración de los controladores fueron determinadas a partir de prueba y error basado en el índice de rendimiento de la Integral del Error Absoluto (IAE). Se realizaron pruebas con distintos valores de calibración y se eligieron las constantes para las cuales el valor IAE fue el más bajo, y que permiten un adecuado funcionamiento del modelo virtual del robot dentro de CoppeliaSim.

Se escogió el índice IAE ya que penaliza y da más importancia a errores pequeños que a errores grandes [32], también tiende a producir una respuesta más lenta que los sistemas óptimos ISE, pero generalmente con una oscilación menos sostenida, lo cual es muy importante para tareas robóticas de alta precisión [33].

El procedimiento realizado es el siguiente:

- Inicialmente se calibran las constantes de los controladores que permiten controlar la posición del efector final del robot manipulador virtual.
- Posteriormente, se calibran las constantes que permiten controlar la orientación del efector final del robot, una vez fijadas las del control de posición.

En la Tabla 3.1 se resumen los parámetros de calibración del controlador PID y el índice IAE obtenido para cada una de las pruebas realizadas. En primer lugar, los parámetros del control de posición, y posteriormente los de orientación.

Tabla 3.1 Pruebas para calibrar los parámetros del controlador PID

Control de Posición				
k_p	0.5	1.5	2.5	3.5
IAE	0.9044	0.4565	0.2511	0.1496
Control de Posición y Orientación				
k_p	2.5	2.5	2.5	2.5
k_o	0.5	1.0	2.0	3.0
IAE	4.29	2.21	1.1072	0.7971

En las Figuras Figura 3.7 y Figura 3.8, se muestran las gráficas del índice de desempeño IAE vs los parámetros de la calibración del controlador PID de acuerdo a las pruebas realizadas.

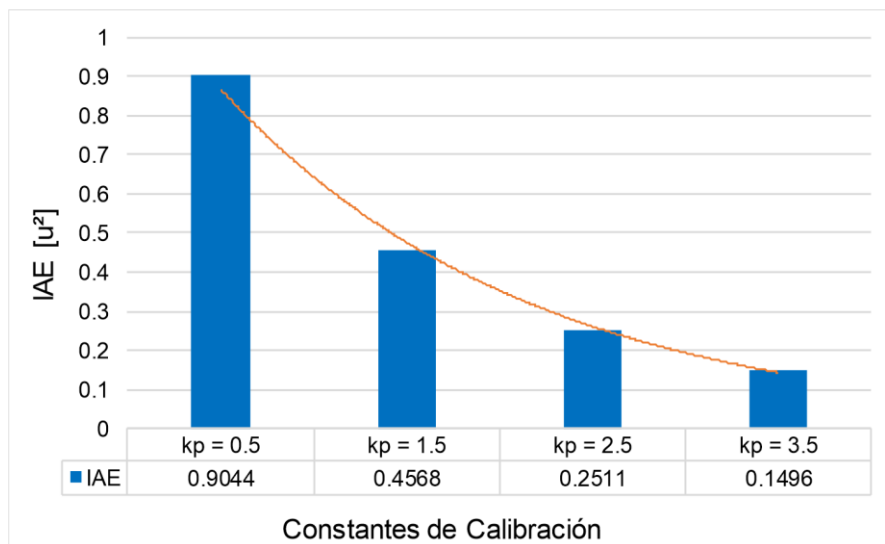


Figura 3.7 Índice de desempeño IAE vs. constantes de calibración del controlador PID para el control de posición del efector final del robot

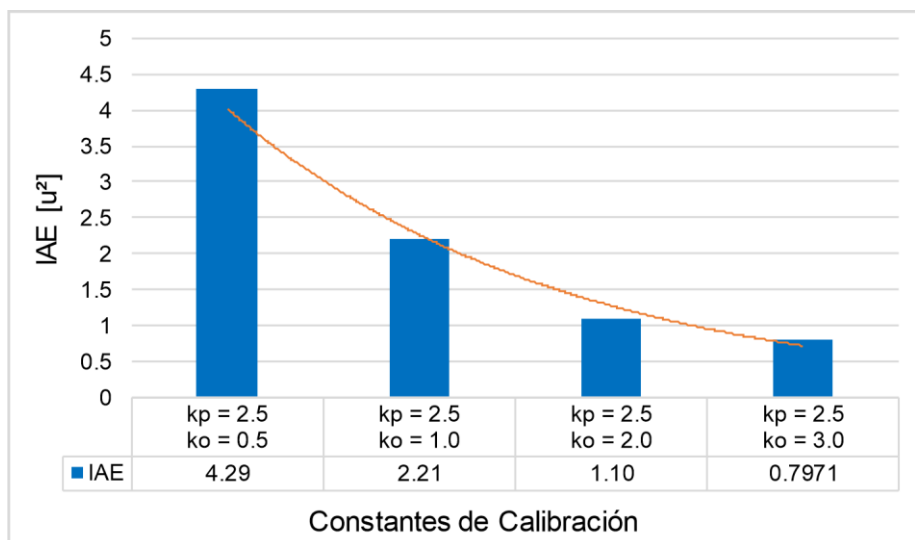


Figura 3.8 Índice de desempeño IAE vs. constantes de calibración del controlador PID para el control de posición y orientación del efector final del robot

En la Figura 3.9 se muestra la evolución de la norma del error del sistema frente a diferentes constantes de calibración del controlador PID, tanto para el control de posición, como para el de orientación del efector final de robot.

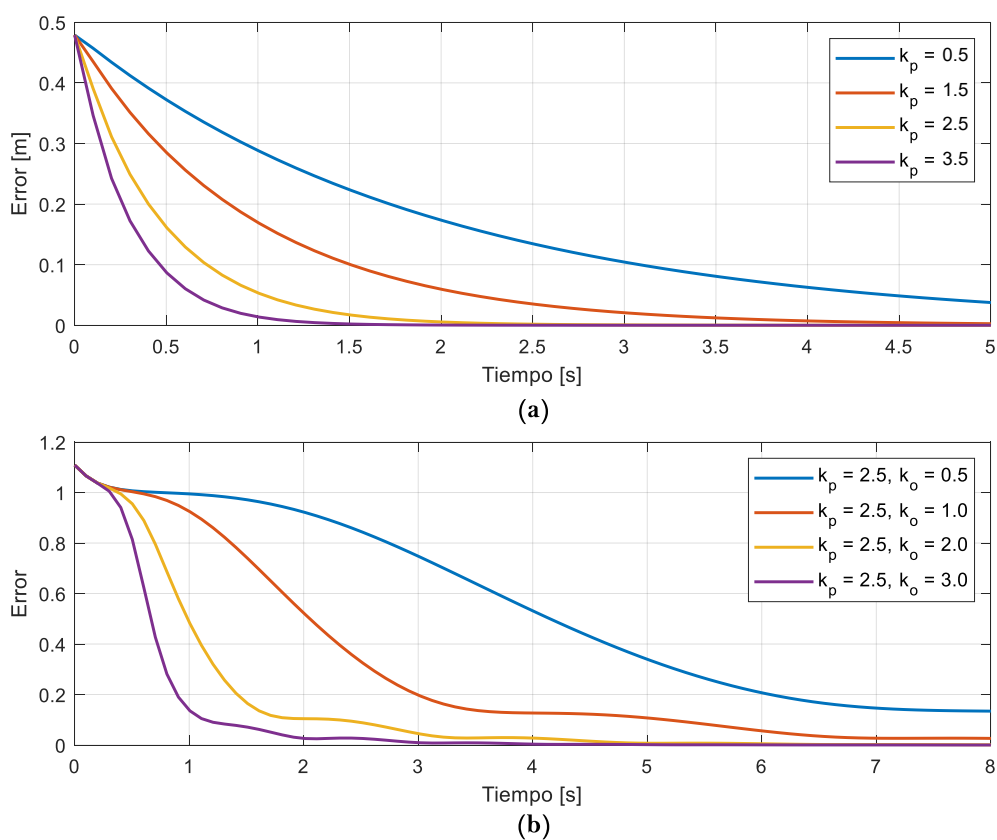


Figura 3.9 Norma del error de posición y orientación del sistema frente a diferentes constantes de calibración a) Control de Posición b) Control de Posición y Orientación

En la Tabla 3.2. se resumen los parámetros de calibración del controlador de álgebra lineal y el índice IAE obtenido para cada una de las pruebas realizadas, tanto para los parámetros del control de posición, como para el de orientación.

Tabla 3.2 Pruebas para calibrar los parámetros del controlador de álgebra lineal (LAC)

Control de Posición				
k_x, k_y, k_z	0.25	0.5	0.75	0.90
IAE	0.0758	0.0925	0.1820	0.3645
Control de Posición y Orientación				
k_x, k_y, k_z	0.80	0.80	0.80	0.80
k_1, k_2, k_3	0.25	0.5	0.75	0.90
IAE	1.1320	0.8425	0.8695	2.1418

En las Figura 3.10 y Figura 3.11 se muestran las gráficas del índice de desempeño IAE vs los parámetros de la calibración del controlador de álgebra lineal de acuerdo a las pruebas realizadas.

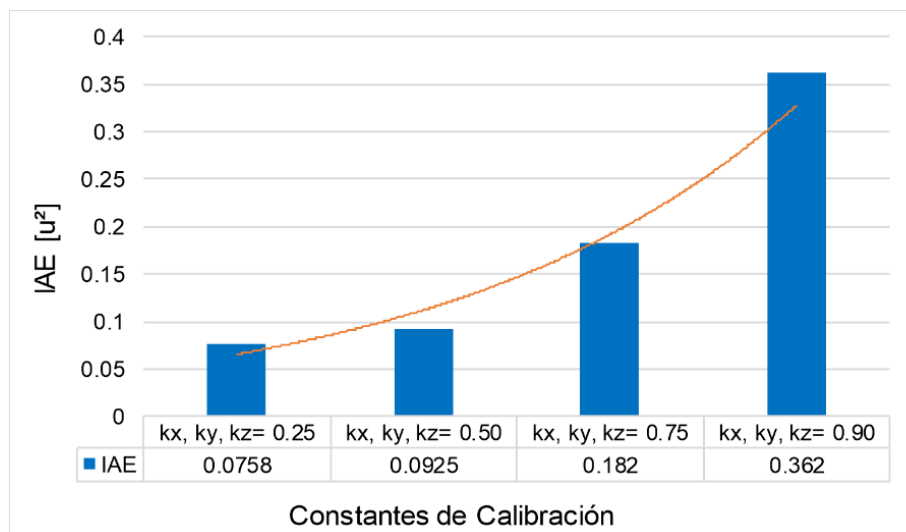


Figura 3.10 Índice de desempeño IAE vs. constantes de calibración del controlador de álgebra lineal para el control de posición del efector final del robot

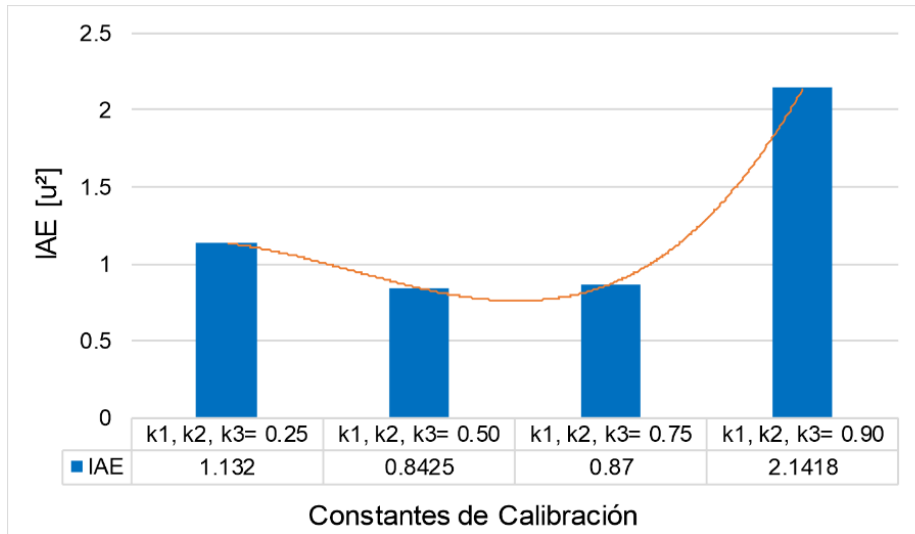


Figura 3.11 Índice de desempeño IAE vs. constantes de calibración del controlador de álgebra lineal para el control de posición y orientación del efector final del robot

En la Figura 3.12 se muestra la evolución de la norma del error del sistema frente a diferentes constantes de calibración del controlador de álgebra lineal. Inicialmente se calibran los parámetros del control de posición, y posteriormente las del control de orientación.

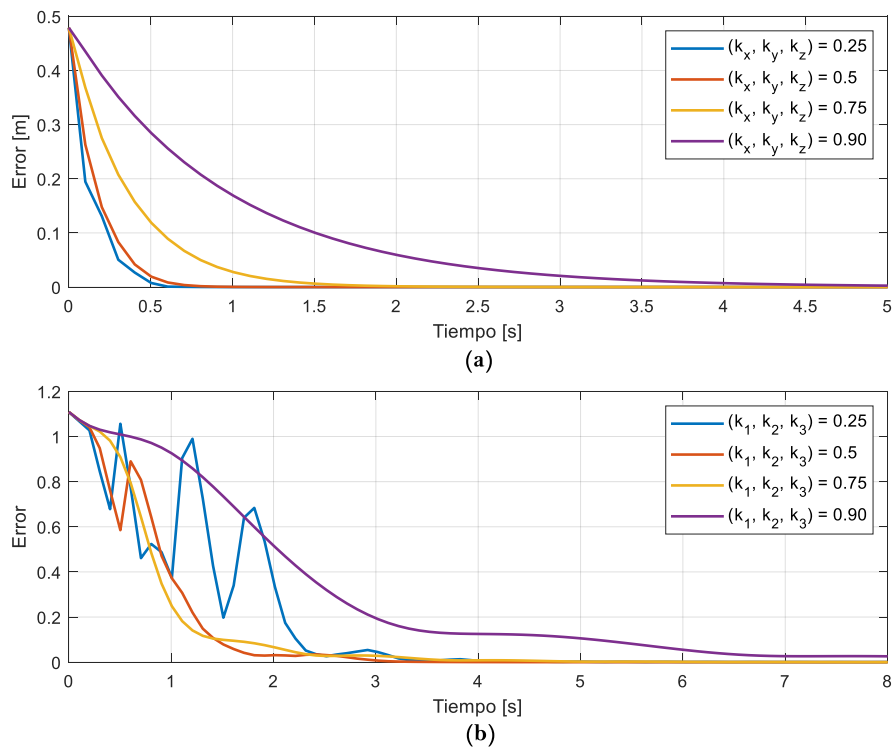


Figura 3.12 Norma del error de posición y orientación del sistema frente a diferentes constantes de calibración a) Control de Posición b) Control de Posición y Orientación

Las constantes de calibración de cada controlador, se presentan en la Tabla 3.3.

Tabla 3.3 Constantes de calibración de los controladores

Controlador	Constante	Valor
PID de Mínima Norma Ecuación (1.38)	K_p	$\begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 2.5 \end{bmatrix}$
	K_o	$\begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix}$
Controlador basado en Álgebra Lineal Ecuación (1.50)	k_x	0.70
	k_y	0.70
	k_z	0.70
	k_1	0.80
	k_2	0.80
	k_3	0.80

3.3 PRUEBAS SIN EL SISTEMA DE RECONOCIMIENTO

3.3.1 CONTROL DE POSICIÓN Y ORIENTACIÓN

El objetivo de esta prueba de funcionamiento es que el efector final del robot virtual UR5 alcance una referencia de posición dada por un punto de coordenadas (x, y, z) dentro de su área de trabajo, y una referencia de orientación, dada por cualquiera de las presentadas en la Sección 2.5, que coinciden con las superficies presentadas en la Figura 3.13a. Por lo tanto, la primera prueba consiste en que el efector final, desde su posición inicial $x_0 = 0.4850 \text{ m}$, $y_0 = -0.10 \text{ m}$, $z_0 = 0.98 \text{ m}$ establecida por su posición de reposo, alcance el punto p_{ref} de coordenadas $x = 0.15 \text{ m}$, $y = 0.15 \text{ m}$, $z = 0.75 \text{ m}$ y una referencia de orientación dirigida hacia la superficie A del entorno virtual, ver Figura 3.13b.

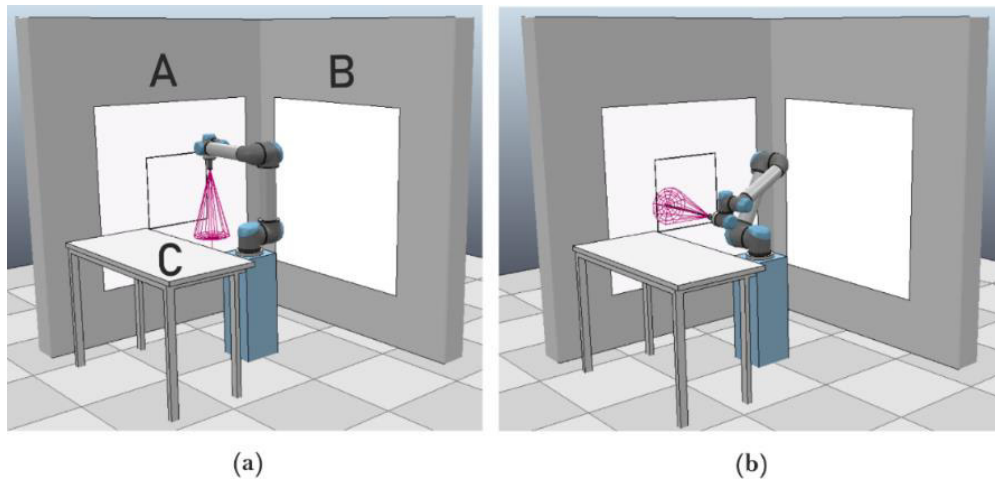


Figura 3.13 Entorno virtual para un manipulador robótico UR5 a) Superficies A, B y C del entorno virtual b) Resultados de la prueba

En la Figura 3.14, se presenta la evolución de la trayectoria que describe el efector final durante esta prueba, donde la referencia es alcanzada por ambos controladores de forma directa, pero poseen un comportamiento ligeramente diferente entre sí para alcanzar la referencia de posición y orientación.

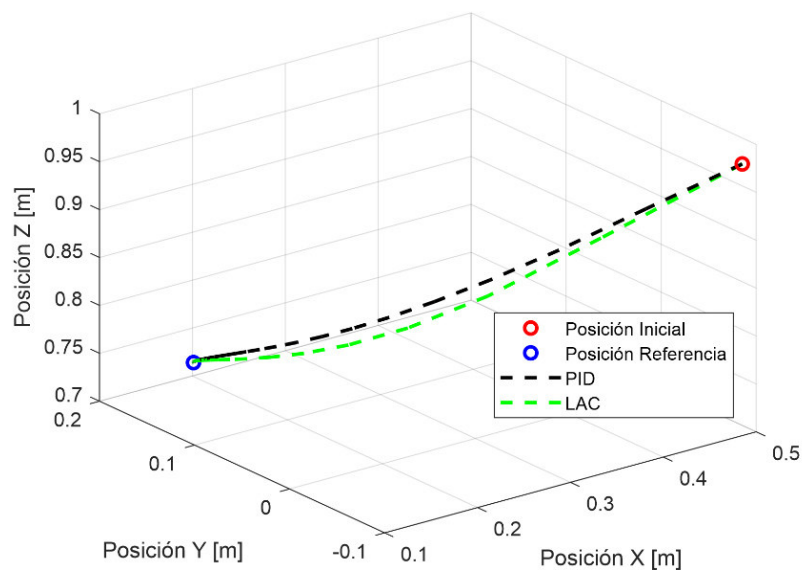


Figura 3.14 Evolución de la posición del efector final para el control de posición y orientación

En la Figura 3.15, se puede observar que el error de posición converge siempre a cero, lo que significa que el robot virtual alcanza su objetivo de control. Adicionalmente, el tiempo de establecimiento para el control de posición para ambos controladores es cercano a los 1.5 y 2 segundos, siendo el controlador basado en álgebra lineal ligeramente más rápido que el controlador PID en alcanzar la referencia de posición.

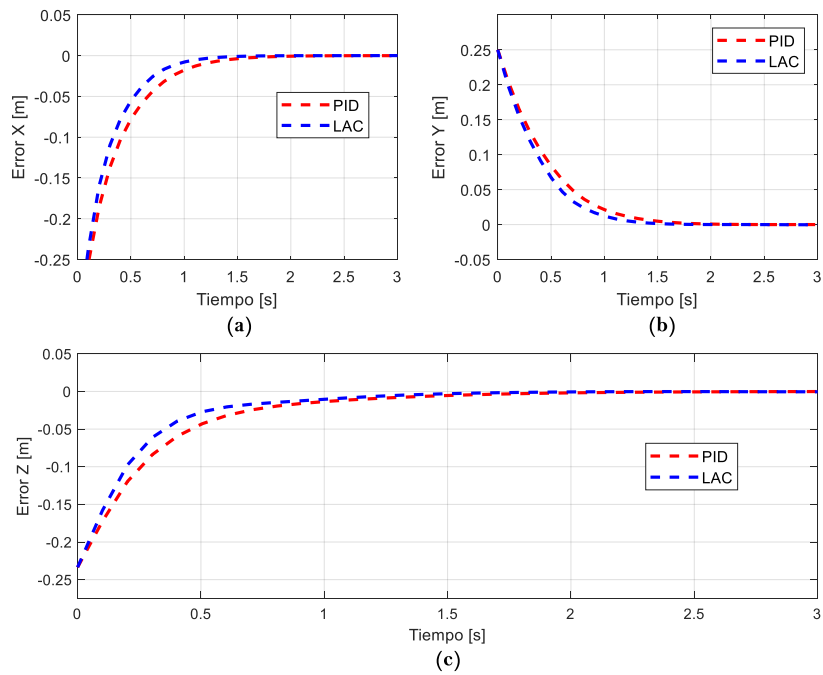


Figura 3.15 Evolución del error de posición a) En el eje X b) En el eje Y c) En el eje Z

En la Figura 3.16, se presenta la norma euclidiana del vector de error de posición y orientación, planteada en la Sección 3.3.1.1, en donde se observa que ambos errores convergen a cero, lo que indica que el efector final alcanzó la referencia establecida. Adicionalmente, el tiempo de establecimiento para alcanzar la referencia de orientación es de aproximadamente 6 segundos para ambos controladores.

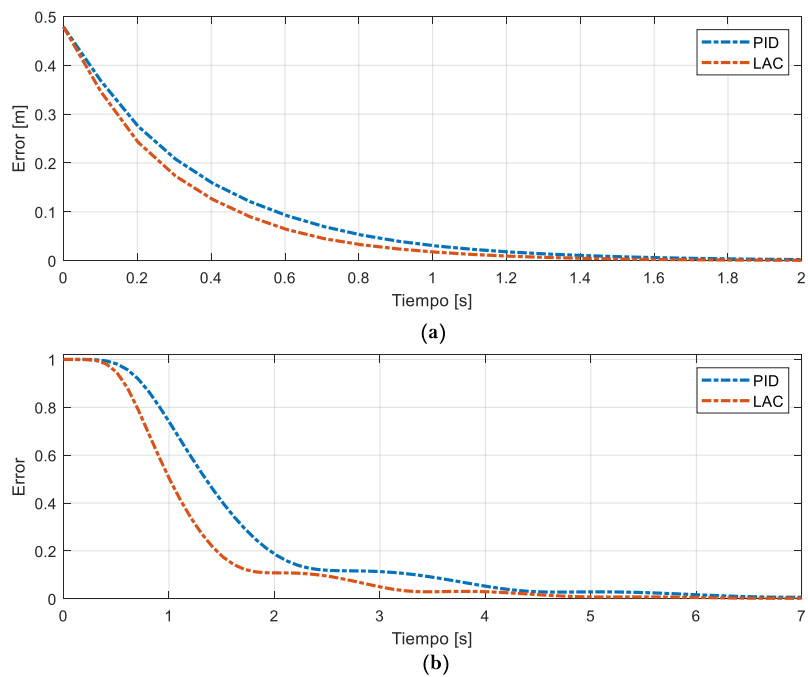


Figura 3.16 Evolución de la norma del error a) Error de posición b) Error de orientación

La segunda prueba consiste en que el efector final alcance el punto p_{ref} de coordenadas $x = 0.15 \text{ m}$, $y = -0.15 \text{ m}$, $z = 0.75 \text{ m}$ y una referencia de orientación dirigida hacia la superficie B del entorno virtual, ver Figura 3.17, partiendo desde su posición de reposo.

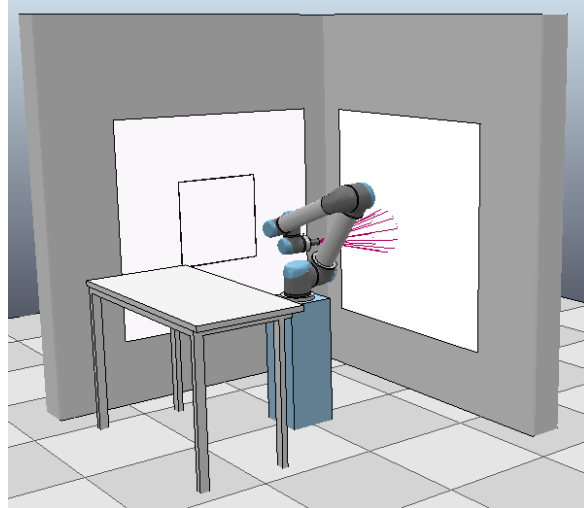


Figura 3.17 Resultados de la segunda prueba del control de posición y orientación

En la Figura 3.18, se presenta la evolución de la trayectoria que describe el efector final durante esta prueba, donde la referencia es alcanzada por ambos controladores rápidamente, siendo ambas respuestas, prácticamente idénticas.

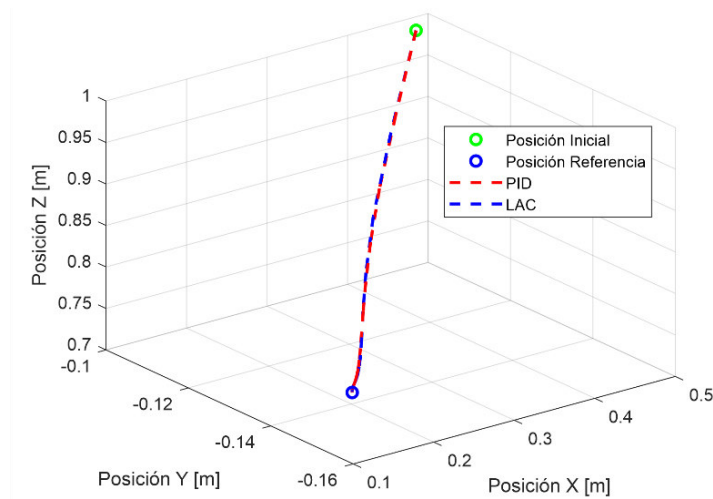


Figura 3.18 Evolución de la posición del efector final para el control de posición y orientación

En la Figura 3.19, se presenta la norma del error de posición y orientación, en donde convergen a cero, lo que indica que el efector final alcanzó sin inconvenientes su referencia. Adicionalmente, el tiempo de establecimiento obtenido es bastante similar para ambos controladores, de aproximadamente 3 o 4 segundos para el control de orientación,

mientras que, para la posición, es de 1 a 1.5 segundos, siendo el controlador de álgebra lineal ligeramente más rápido que el controlador PID en alcanzar ambas referencias.

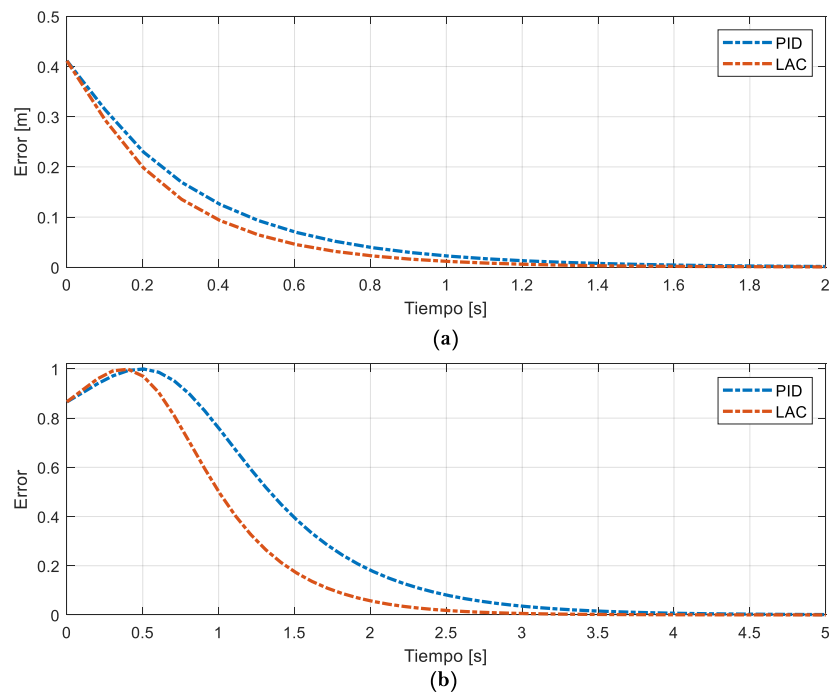


Figura 3.19 Evolución de la norma del error del sistema a) Posición b) Orientación

A continuación, se presenta el análisis del rendimiento de los controladores implementados para las pruebas del control de posición, utilizando el índice de rendimiento IAE.

3.3.1.1 Rendimiento de los Controladores

El índice de rendimiento utilizado es la Integral del Error Absoluto (IAE), y se calcula a partir de la siguiente ecuación:

$$IAE_{e,o} = \int_0^{\infty} |e_{p,o}(t)| dt \quad (3.1)$$

El valor total del criterio se calcula obteniendo la norma euclidiana del vector IAE calculada a partir de la evolución de los errores de posición e_p y orientación e_o en el tiempo. Por lo que, tanto para el control de posición, como para los demás modos de funcionamiento del robot, el valor total del índice IAE se define como:

$$IAE = \|\|IAE_{p,o}\|\| \quad (3.2)$$

En la Tabla 3.4 se presenta el índice de rendimiento IAE para cada uno de los controladores utilizados en la pruebas del control de posición.

Tabla 3.4 Índice de rendimiento IAE de cada controlador para el control de posición

IAE PARA EL CONTROL DE POSICIÓN	Prueba	Controladores	
		PID	LAC
	1	2.2811	1.944
2	1.412	1.171	

En la Tabla 3.5 se presentan los tiempos de establecimiento de las respuestas de cada uno de los controladores utilizados en las pruebas del control de posición.

Tabla 3.5 Tiempo de establecimiento de cada controlador para el control de posición

TIEMPO DE ESTABLECIMIENTO	Pruebas	Posición		Orientación	
		PID	LAC	PID	LAC
	1	2 s	1.5 s	6.5 s	6.5 s
2	1.25 s	1.5 s	2.5 s	3.5 s	

A partir de los resultados de la Tabla 3.4 y Tabla 3.5, se puede observar que el controlador basado en álgebra lineal es ligeramente superior al controlador PID ya que su índice IAE es menor en todas las pruebas realizadas. Sin embargo, ambos cumplen con el objetivo de control, además, ninguno presenta sobre impulsos en sus respuestas, y sus tiempos de establecimiento son prácticamente idénticos.

3.3.2 SEGUIMIENTO DE TRAYECTORIAS

El objetivo de control de esta prueba de funcionamiento es que el efector final del robot virtual UR5 siga dos tipos de trayectorias, cuadrada y circular, manteniendo una orientación específica, que puede ser en dirección de cualquiera de las superficies mostradas en la Figura 3.13. Las ecuaciones matemáticas que describen las trayectorias a seguir se muestran en la Tabla 3.6.

Tabla 3.6 Trayectorias propuestas

Tray.	Componentes		
	Eje X	Eje Y	Eje Z
Cuad.	$f(t) = X_c$	$\begin{cases} Y_c + \frac{L}{2} & t \leq T/4 \\ Y_c + \frac{L}{2} - \frac{L(t-T/4)}{(T/4)} & T/4 \leq t \leq T/2 \\ Y_c - \frac{L}{2} & T/2 \leq t \leq 3T/4 \\ Y_c - \frac{L}{2} + \frac{L(t-3T/4)}{(T/4)} & t \geq 3T/4 \end{cases}$	$\begin{cases} Z_c - \frac{L}{2} + \frac{L}{(T/4)} & t \leq T/4 \\ Z_c + \frac{L}{2} & T/4 \leq t \leq T/2 \\ Z_c + \frac{L}{2} - \frac{L(t-T/2)}{(T/4)} & T/2 \leq t \leq 3T/4 \\ Z_c - \frac{L}{2} & t \geq 3T/4 \end{cases}$
Circ.	$f(t) = R \sin\left(\frac{2\pi t}{T}\right) + X_c$	$f(t) = R \cos\left(\frac{2\pi t}{T}\right) + Y_c$	$f(t) = Z_c$

Donde R es el radio de la trayectoria circular, L es la longitud de un lado de la trayectoria cuadrada, y (X_c, Y_c, Z_c) son las coordenadas del centroide de la trayectoria a dibujar, y T es el tiempo de duración de la trayectoria de referencia para completar su recorrido.

La primera prueba consiste en seguir una trayectoria cuadrada, cuyo centroide p_{centro} tiene las siguientes coordenadas $X_c = 0.40\text{ m}$, $Y_c = 0\text{ m}$, $Z_c = 0.80\text{ m}$, L es igual a 40 cm , tiene una duración T de 40 segundos , y una referencia de orientación dirigida hacia la superficie A del entorno virtual, ver Figura 3.20a.

La segunda prueba consiste en seguir una trayectoria circular, cuyo centroide p_{centro} tiene las siguientes coordenadas $X_c = 0\text{ m}$, $Y_c = 0.55\text{ m}$, $Z_c = 0.80\text{ m}$, R igual a 20 cm , una duración T de 40 segundos , y una referencia de orientación dirigida hacia la superficie C del entorno virtual, ver Figura 3.20b.

Los resultados en CoppeliaSim de estas pruebas de funcionamiento del seguimiento de ambas trayectorias de referencia se presentan en la Figura 3.20, en donde se utilizó la pistola de pintura o *paint gun* durante toda su extensión para evidenciar de mejor manera los resultados.

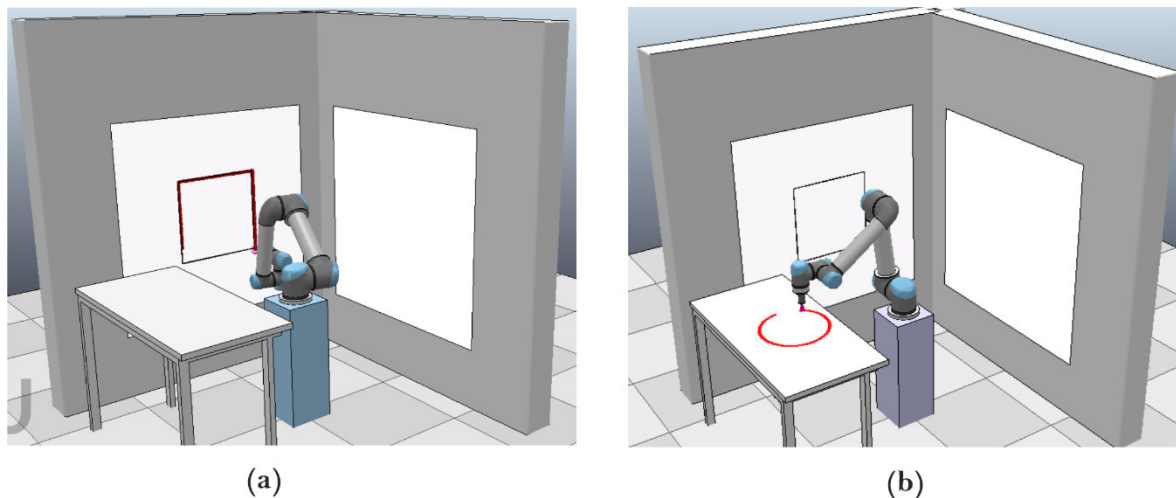


Figura 3.20 Simulación del seguimiento de trayectorias a) Trayectoria Cuadrada b) Trayectoria Circular

3.3.2.1 Trayectoria Cuadrada

En la Figura 3.21 y Figura 3.22 se puede observar la evolución de la posición del efector final durante el seguimiento de la trayectoria cuadrada para ambos controladores, en donde el sistema alcanza la trayectoria a lo largo de toda su extensión partiendo desde su posición inicial $x_0 = 0.4850\text{ m}$, $y_0 = -0.10\text{ m}$, $z_0 = 0.98\text{ m}$.

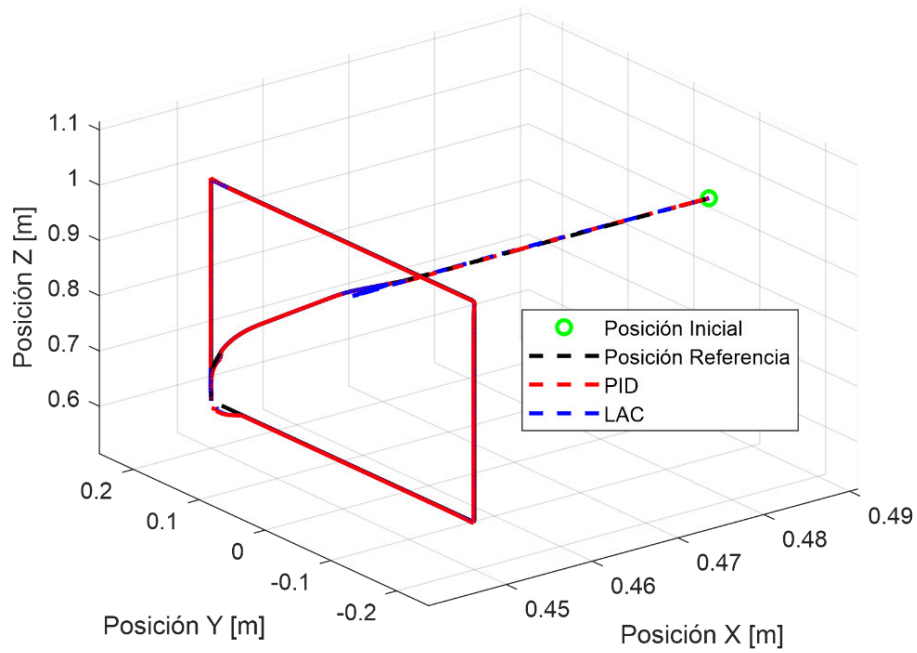


Figura 3.21 Evolución de la posición del efector final para el seguimiento de una trayectoria cuadrada

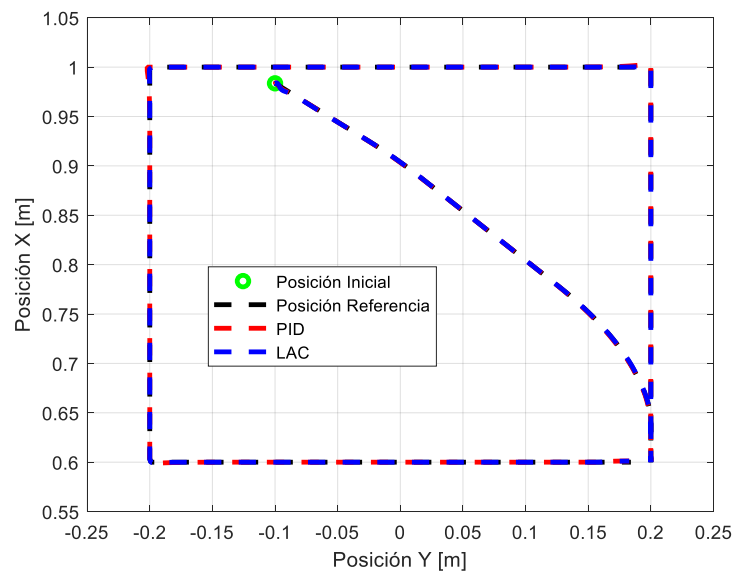


Figura 3.22 Evolución de la posición del efector final para el seguimiento de una trayectoria cuadrada en el plano Y-Z

En la Figura 3.23 se puede observar como ambos controladores presentan pequeños errores durante el inicio de la prueba de simulación, es decir, durante su etapa transitoria, pero una vez alcanzada la señal de referencia, su error de posición se mantiene en cero.

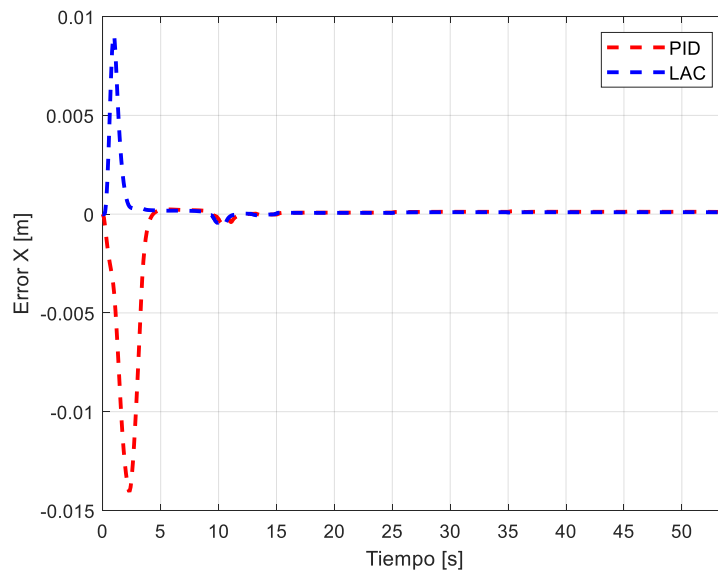


Figura 3.23 Error de seguimiento en el eje X de una trayectoria cuadrada

En la Figura 3.24 y Figura 3.25, se presentan las respuestas del sistema controlado por el PID, en donde se puede observar el error obtenido durante el inicio de la prueba de funcionamiento, que no supera los 1.6 cm, y corresponde al seguimiento de la trayectoria auxiliar utilizada para que el efector final alcance la referencia de la trayectoria principal, adicionalmente, durante esta misma etapa, también se está controlando la orientación del efector final lo cual impide que el error de posición se estabilice en cero rápidamente. Sin embargo, al utilizar el controlador de álgebra lineal, éste es capaz de mantenerse y estabilizarse en cero rápidamente.

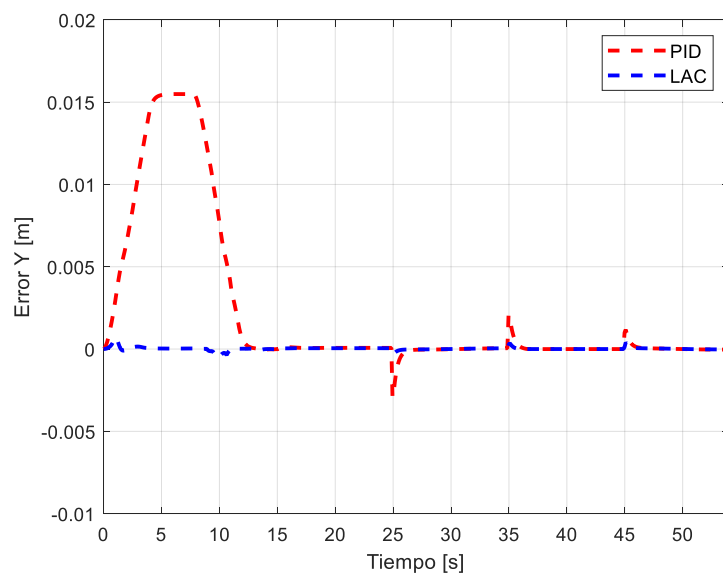


Figura 3.24 Error de seguimiento en el eje Y de una trayectoria cuadrada

En la Figura 3.24 y Figura 3.25 se puede observar pequeños sobrepicos causados por los cambios bruscos provocados por la trayectoria de referencia cada vez que alcanza una esquina de su forma, éstos se dan en los segundos 25, 35 y 45.

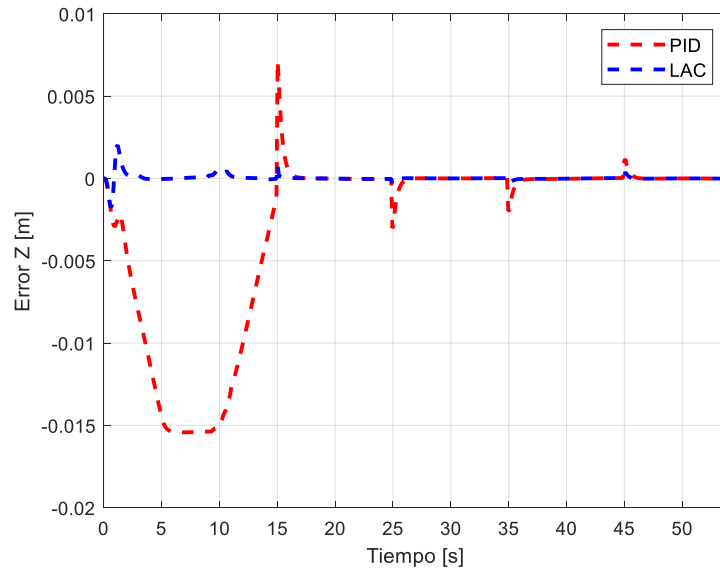


Figura 3.25 Error de seguimiento en el eje Z de una trayectoria cuadrada

En todos los casos, tanto el error de posición como el de orientación convergen a cero, ver Figura 3.26. Adicionalmente, el tiempo de establecimiento para el control de orientación es cercano a los 7 segundos en ambos controladores, en donde una vez alcanzada la referencia, el error se mantiene nulo durante el resto de la etapa de seguimiento.

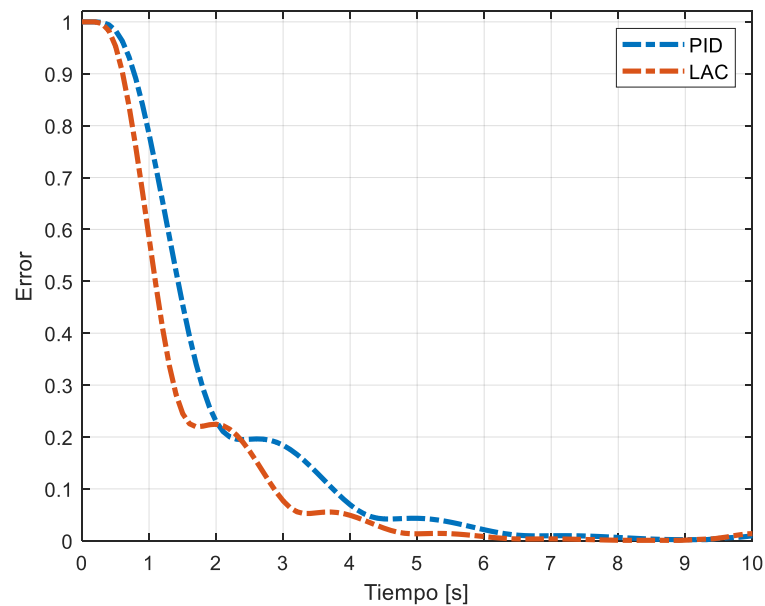


Figura 3.26 Evolución de la norma del error de orientación

El controlador de álgebra lineal (LAC) presenta un mejor desempeño en los cambios bruscos de la referencia cuadrada, ya que el PID presenta un pequeño sobreimpulso de aproximadamente el 1% en cada esquina de la figura, ligeramente superior al obtenido con el LAC, que aumenta según se aumente la velocidad de referencia del efector final.

3.3.2.2 Trayectoria Circular

En la Figura 3.27 y Figura 3.28 se puede observar la evolución de la posición del efector final durante el seguimiento de la trayectoria circular para ambos controladores. En ambos casos, el sistema alcanza la trayectoria de forma satisfactoria.

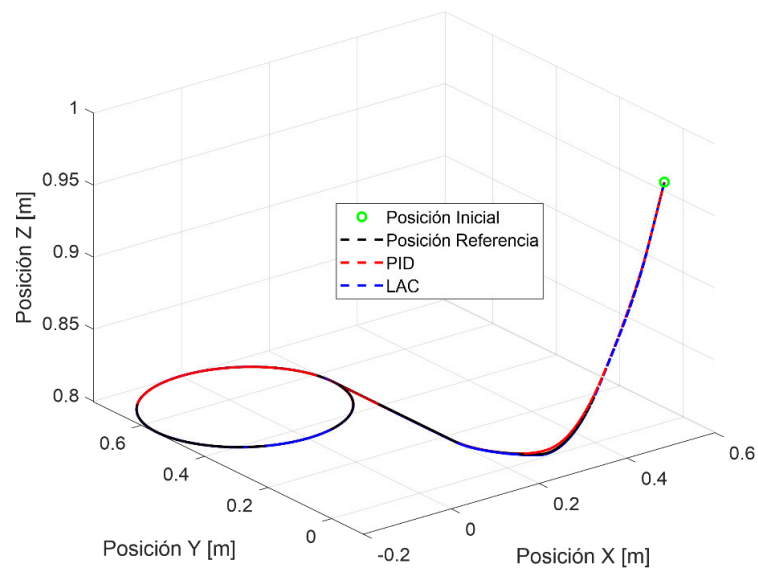


Figura 3.27 Evolución de la posición del efector final para el seguimiento de una trayectoria circular

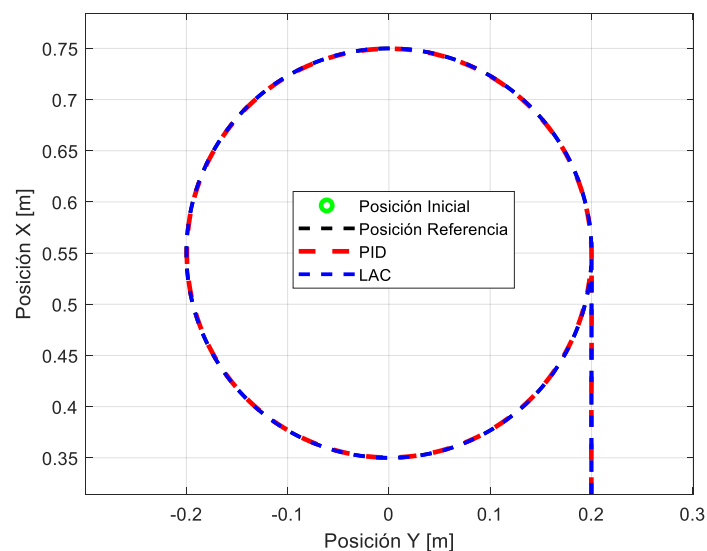


Figura 3.28 Evolución de la posición del efector final para el seguimiento de una trayectoria circular en el plano X-Y

En los primeros instantes de los resultados presentados en la Figura 3.29, Figura 3.30 y Figura 3.31 se puede observar el error del sistema controlado por el PID, que corresponde al seguimiento de la trayectoria auxiliar. Posteriormente, a partir del segundo 15, la respuesta del controlador PID presenta un error de seguimiento que posee un comportamiento sinusoidal con una amplitud de no más de 1.5 cm alrededor de la referencia de posición puesto que es de naturaleza circular, y que son mucho más evidentes en los ejes y y z , que en x , puesto que la referencia de posición en este eje, es constante. Por otra parte, el controlador LAC mantiene prácticamente en cero tanto el error de posición, como el de orientación a lo largo de la trayectoria auxiliar y circular.

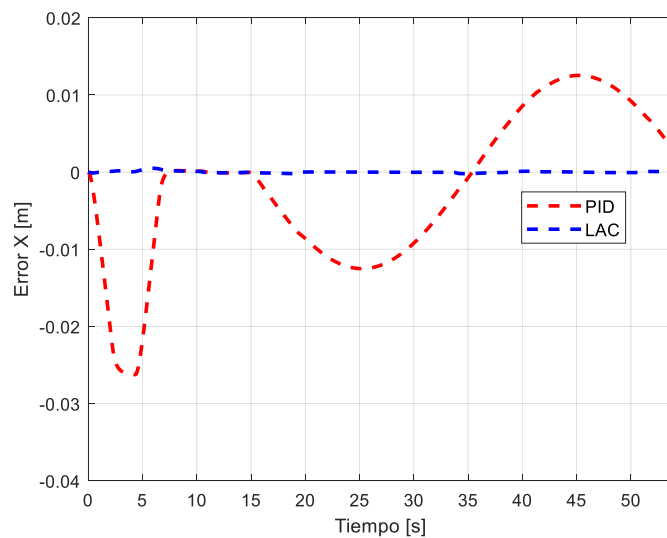


Figura 3.29 Error de seguimiento en el eje X de una trayectoria circular

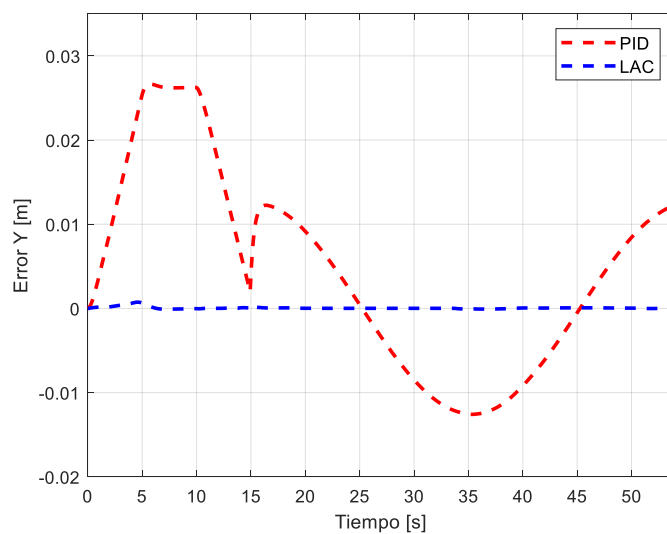


Figura 3.30 Error de seguimiento en el eje Y de una trayectoria circular

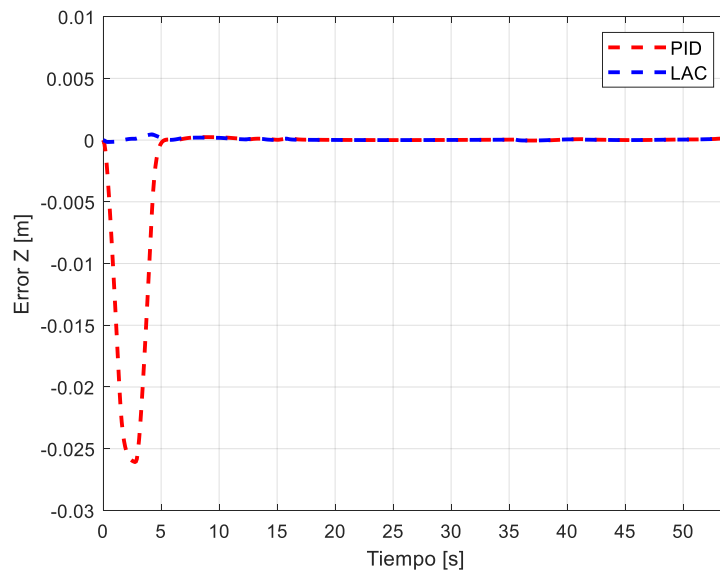


Figura 3.31 Error de seguimiento en el eje Z de una trayectoria circular

Los controladores mantuvieron un error nulo a lo largo de toda la trayectoria circular, es decir, se mantuvo la orientación del efector final hacia la superficie C en todo momento.

3.3.2.3 Rendimiento de los Controladores

En la Tabla 3.7 se presenta el índice de rendimiento IAE para cada uno de los controladores utilizados en la prueba de seguimiento de trayectorias, tanto para la trayectoria cuadrada, como para la circular. Sin embargo, a diferencia del control de posición, no se incluye su tiempo de establecimiento, ya que, en este caso, la trayectoria auxiliar y principal parten de forma lenta y continua desde la posición inicial del efector final. Además, tampoco se incluye el tiempo de establecimiento en la etapa de control de orientación para la trayectoria circular, puesto que el efector final en su estado inicial ya parte con esa orientación de referencia, y su error se mantiene nulo durante toda la prueba de funcionamiento para ambos controladores.

Tabla 3.7 Índice de rendimiento IAE para cada controlador y trayectoria

IAE PARA EL SEGUIMIENTO DE TRAYECTORIAS	Trayectoria	Controladores	
		PID	LAC
	Cuadrada	1.793	1.546
Circular	0.658	0.09	

En la Tabla 3.8 se presentan los tiempos de establecimiento de las respuestas de cada uno de los controladores para su etapa de control de orientación.

Tabla 3.8 Tiempo de establecimiento de cada controlador para el seguimiento de trayectorias

TIEMPO DE ESTABLECIMIENTO	Pruebas	Orientación	
		PID	LAC
	Cuadrada	6 s	6.5 s

A partir de los resultados de la Tabla 3.7 y Tabla 3.8, se puede observar que el controlador LAC es ligeramente superior al controlador PID. Sin embargo, ambos cumplen con su objetivo de control con un tiempo de establecimiento prácticamente idéntico, y tanto el error de posición como de orientación convergen a cero rápidamente.

3.4 PRUEBAS CON EL SISTEMA DE RECONOCIMIENTO

3.4.1 MODO HGR+IMU

En esta prueba de funcionamiento se utiliza el modelo general del sistema de reconocimiento de gestos desarrollado en el Proyecto de Investigación PIGR-19-07, cuyas características se indican en la Sección 1.3.5.3, en conjunto con la IMU del sensor Myo Armband, cuyo funcionamiento en conjunto se explica a detalle en la Sección 2.5 del capítulo anterior. En la Figura 3.32, se muestra su funcionamiento al ser utilizada por un usuario con su brazo derecho.

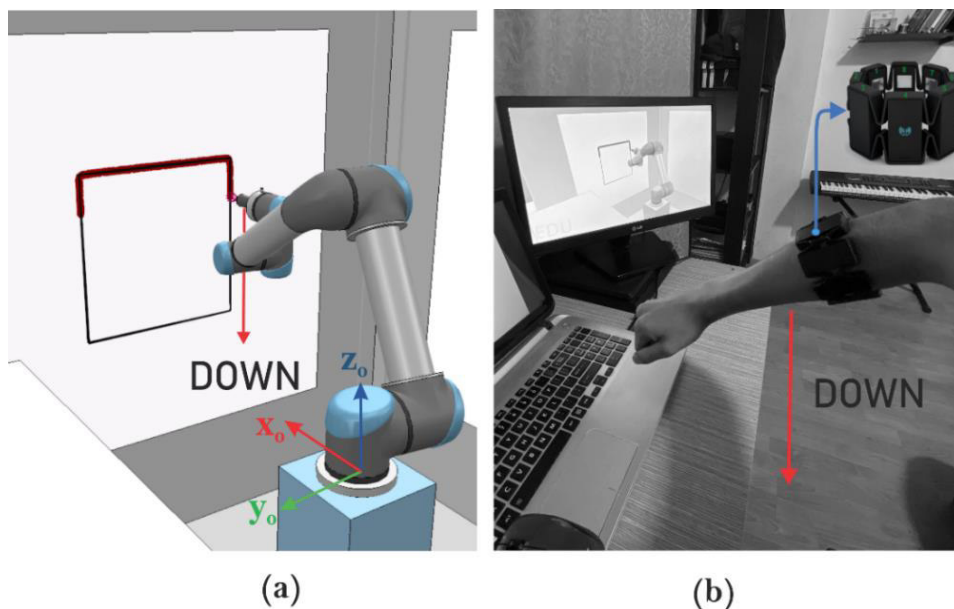


Figura 3.32 Manipulador robótico UR5 controlado a través de la interfaz humano-máquina basada en el sensor Myo Armband

Esta prueba de funcionamiento consiste en pintar una trayectoria cuadrada en la superficie A del entorno virtual, ver Figura 3.13, mediante la creación de perfiles de referencia para el

efector final del robot a partir de los comandos de selección generados a partir del sistema de reconocimiento de gestos y comandos de movimiento generados a partir de la IMU del sensor Myo Armband, los cuales serán utilizados como entradas de referencia de los controladores diseñados. Las trayectorias a dibujar tienen las mismas dimensiones que la prueba de seguimiento de trayectorias presentada en la Sección 3.3.2.1.

En la Figura 3.33 se puede observar la evolución de la posición del efector final del robot frente a las referencias de posición, *Referencia 1* y *Referencia 2*, generadas por un usuario al utilizar la interfaz humano-máquina propuesta constituida por el sistema de reconocimiento de gestos y la IMU del sensor Myo Armband. La *Referencia 1* se utiliza como entrada del controlador PID, mientras que la *Referencia 2* es utilizada por el controlador de álgebra lineal.

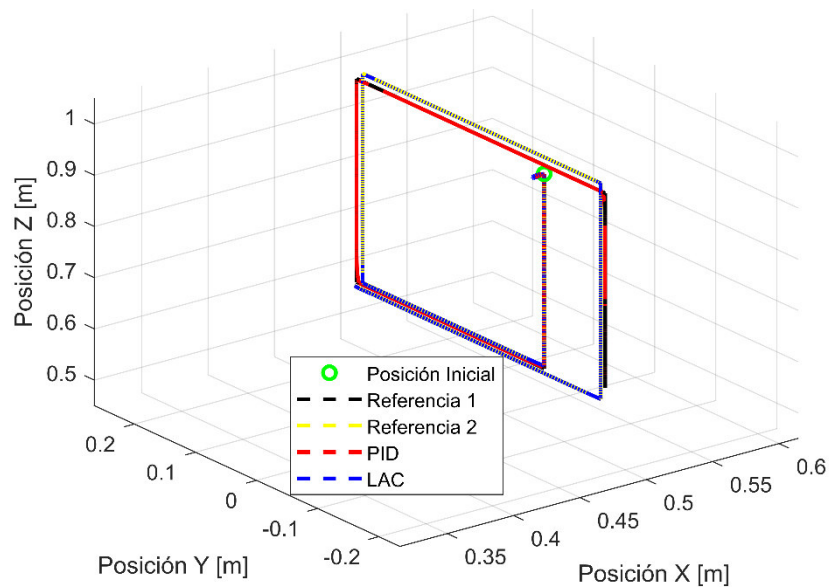


Figura 3.33 Evolución de la posición del efector final frente a los perfiles Referencia 1 y 2 generados con la interfaz humano-máquina del modo HGR+IMU

En la Figura 3.34 se puede observar el plano Y-Z del entorno virtual donde se puede visualizar tanto la referencias generadas por la interfaz, como la evolución de la posición del efector final durante estas pruebas para ambos controladores.

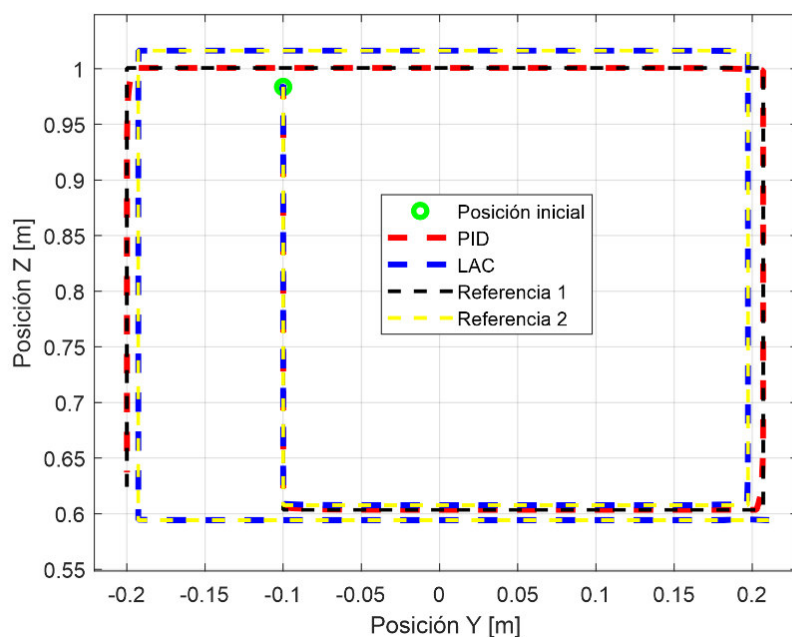


Figura 3.34 Evolución de la posición del efector final en el plano Y-Z frente a los perfiles Referencia 1 y 2 generados con la interfaz humano-máquina

En la Figura 3.35, 3.36 y 3.37 se puede observar que el error de posición converge siempre a cero, lo que significa que el robot virtual siempre sigue las referencias generadas con la interfaz humano-máquina utilizada.

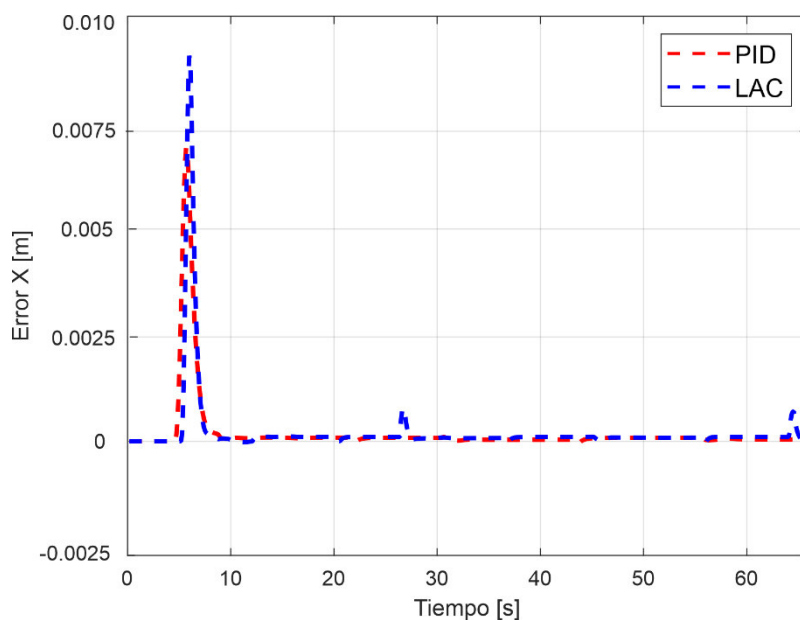


Figura 3.35 Error de seguimiento en el eje X

En la Figura 3.36 y Figura 3.37 se pueden observar pequeños errores de seguimiento de no más de 2.5 cm, que se generan debido al cambio de dirección que se da al formar la

trayectoria cuadrada con los comandos de movimiento de la interfaz. Sin embargo, el error siempre tiende a ser nulo en ambos controladores.

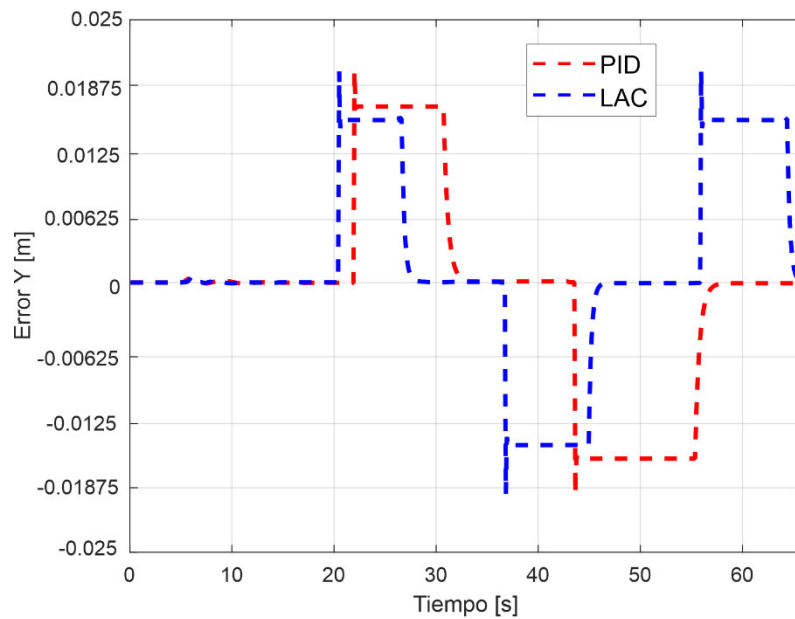


Figura 3.36 Error de seguimiento en el eje Y

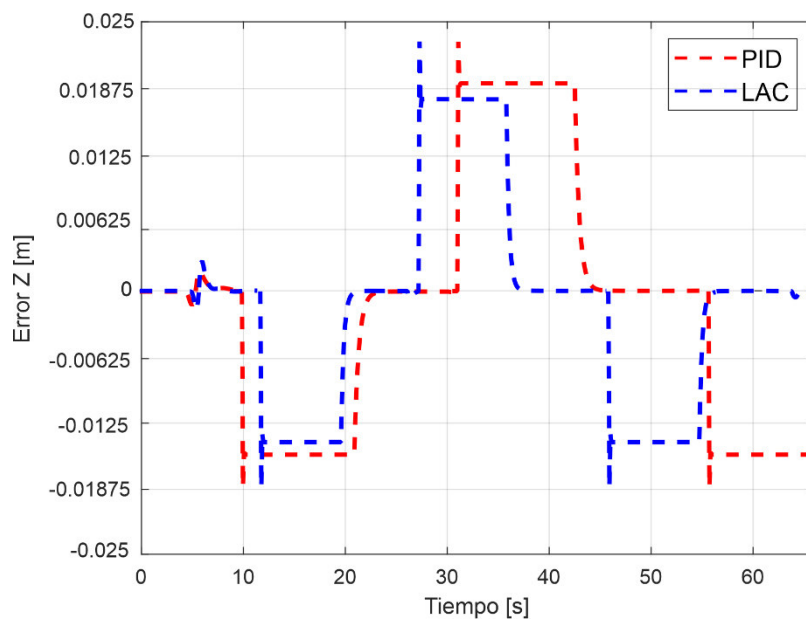


Figura 3.37 Error de seguimiento en el eje Z

En la Figura 3.38 se presenta la norma euclidiana del vector de error de orientación, en donde se observa que converge a cero en aproximadamente 6 segundos, en ambos controladores, una vez cambiada su referencia de orientación a través de los gestos clasificados por el sistema. Esto indica que el efector final alcanzó la referencia de

orientación establecida para esta prueba de control, y mantuvo esa orientación hasta que finalice la prueba de funcionamiento.

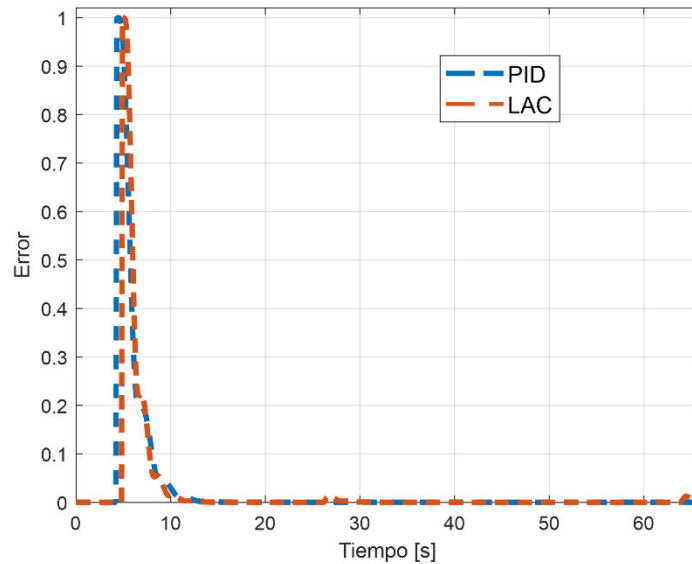


Figura 3.38 Evolución de la norma del error de orientación

3.4.1.1 Rendimiento de los Controladores

En la Tabla 3.9 se presenta el índice de rendimiento IAE para cada uno de los controladores utilizados en la prueba de funcionamiento de la interfaz humano-máquina implementada.

Tabla 3.9 Índice de rendimiento IAE para cada controlador

Trayectoria	Controladores	
	PID	LAC
Interfaz Humano-Máquina	1.760	1.350

A partir de los resultados de la Tabla 3.9, se puede observar que ambos controladores cumplen con su objetivo de control, además, ninguno presenta sobre impulsos en sus respuestas, por lo que su funcionamiento fue satisfactorio.

Finalmente, cabe resaltar que el índice IAE obtenido en esta prueba es ligeramente inferior al obtenido en la prueba de seguimiento de trayectorias, esto debido a que a través de la interfaz humano-máquina se realiza tanto la tarea de seguimiento como de control de orientación una tras otra de forma independiente, y no de forma conjunta como en la prueba anterior, lo que causa un aumento en el índice de rendimiento IAE.

4 CONCLUSIONES Y RECOMENDACIONES

Se resalta que en base al presente trabajo de titulación, se realizó un paper que fue aceptado y presentado en la conferencia ETCM 2021, cuya sede fue en la Universidad de Cuenca, en la rama de *Robotics and Automation Systems*, el miércoles 13 de octubre del 2021, y que ya se encuentra publicado en la página web de IEEE Xplore, cuyo enlace se presenta en [34].

4.1 CONCLUSIONES

- Se implementó el modelo cinemático directo y la matriz Jacobiana del manipulador robótico UR5, los cuales fueron utilizados dentro del sistema de control propuesto en ambas técnicas de control, una vez que las mismas se validaron mediante su simulación en lazo abierto con la ayuda de Matlab y CoppeliaSim.
- Se estableció la diferencia e importancia de utilizar la matriz Jacobiana analítica sobre la geométrica en el control de posición y orientación del efector final de manipuladores robóticos de n grados de libertad.
- Se integró el sistema de reconocimiento de gestos desarrollado en el Proyecto de Investigación PIGR-19-07 con el sistema de control implementado para manejar el manipulador robótico virtual UR5, en donde se pudo evidenciar, a través de las pruebas realizadas, ventajas útiles tales como reconocer gestos complejos de las manos, uso de sensores no invasivos, cómodos de usar, sin conflictos al utilizar la mano (mano y muñeca totalmente libres), y aprovechar la versatilidad del movimiento del antebrazo humano.
- Se evaluó el desempeño de la interfaz humano-máquina propuesta reconociendo los gestos: FIST, WAVEIN, WAVEOUT, OPEN y PINCH, y utilizando las señales de la IMU del Myo Armband para controlar un manipulador robótico en su área de trabajo al generar una trayectoria cuadrada como referencia de posición y orientación, en donde se obtuvieron resultados satisfactorios, y se pudo apreciar las ventajas de este tipo de interfaz, tales como, prescindir del uso de ecuaciones matemáticas para la generación de trayectorias de referencia y el nivel bajo de dificultad que requiere para aprender a usarse. Sin embargo, los comandos de acción y referencias generadas con ella dependen ampliamente de la destreza del usuario que la utilice y cuya dificultad va ligada con la complejidad de la aplicación a realizar.

- Se estableció una vía de comunicación entre Matlab y CoppeliaSim a través de una Interfaz de Programación de Aplicaciones (API) que permitió que el sistema de control diseñado para el manipulador robótico UR5 virtual funcione correctamente, tanto en la adquisición, como en la recepción de datos.
- Se diseñó un entorno virtual de un ambiente controlado simulando condiciones reales a las que se podría enfrentar este tipo de plataforma robótica. Por lo tanto, se sentó las bases necesarias para que estas aplicaciones sirvan de recurso didáctico en temas relacionados con la simulación y control de plataformas robóticas virtuales.
- Para interactuar y visualizar el comportamiento del manipulador robótico UR5 virtual, se desarrolló una interfaz gráfica utilizando la herramienta App Designer de Matlab, siendo ésta, una parte fundamental dentro de la interfaz humano-máquina implementada con el sistema de reconocimiento de gestos, el robot virtual y sus controladores, puesto que brinda al usuario herramientas tanto de control, como de monitoreo intuitivas de manejar.
- En este trabajo se presentó el diseño y comparación de dos controladores enfocados en el control de posición y seguimiento de trayectorias del manipulador robótico UR5, en donde ambos controladores mostraron un rendimiento satisfactorio en todas las pruebas realizadas, ya que su error de posición y orientación siempre convergieron a cero en un tiempo menor a cinco segundos en promedio, y con un índice de desempeño IAE menor a dos unidades en la mayor parte de las pruebas de funcionamiento a pesar de estar diseñados únicamente en base al modelo cinemático del robot.
- Se evidenció que el controlador de álgebra lineal es ligeramente superior a la técnica de control PID de mínima norma debido a su buen desempeño frente a cambios abruptos presentados en las trayectorias de referencia utilizadas.

4.2 RECOMENDACIONES

- Se recomienda obtener el modelo dinámico del robot manipulador y utilizarlo dentro de técnicas de control más robustas con el fin de mejorar las características del sistema para que pueda ser utilizado en un rango más amplio de aplicaciones de mayor complejidad, que requieran control de fuerza y velocidad, y no limitarse únicamente a aplicaciones sencillas con velocidades bajas.
- Se recomienda ajustar las constantes de calibración de cada controlador de acuerdo a un índice de desempeño, para encontrar los valores que permitan que el controlador

presente el mejor rendimiento y comportamiento frente a perturbaciones y cambios de referencia.

- Como trabajo futuro se podría implementar los controladores diseñados en manipuladores robóticos reales, donde además se podría implementar un sistema que permita seguir en línea los movimientos de un brazo humano, ya sea con un sensor Kinect, o con varias unidades de medición inercial, con el objetivo de utilizarse como entradas de referencia adicionales del sistema de control del brazo robótico.

5 REFERENCIAS BIBLIOGRÁFICAS

- [1] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, vol. 118, Springer, 2017.
- [2] A. Oña, V. Vimos, M. Benalcázar y P. J. Cruz, "Adaptive Non-linear Control for a Virtual 3D Manipulator," de *2020 IEEE ANDESCON*, 2020.
- [3] J. Kofman, X. Wu, T. J. Luu y S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE transactions on industrial electronics*, vol. 52, p. 1206–1219, 2005.
- [4] C. Pillajo y J. E. Sierra, "Human Machine Interface HMI using Kinect sensor to control a SCARA Robot," de *2013 IEEE Colombian conference on communications and computing (COLCOM)*, 2013.
- [5] L. I. Barona López, Á. L. Valdivieso Caraguay, V. H. Vimos, J. A. Zea, J. P. Vásquez, M. Álvarez y M. E. Benalcázar, "An Energy-Based Method for Orientation Correction of EMG Bracelet Sensors in Hand Gesture Recognition Systems," *Sensors*, vol. 20, p. 6327, 2020.
- [6] P. Leica, O. Camacho, S. Lozada, R. Guamán, D. Chávez y V. H. Andaluz, "Comparison of control schemes for path tracking of mobile manipulators," *International Journal of Modelling, Identification and Control*, vol. 28, p. 86–96, 2017.
- [7] F. Reyes, *Robótica-control de robots manipuladores*, Alfaomega grupo editor, 2011.
- [8] A. O. Baturone, *Robótica: manipuladores y robots móviles*, Marcombo, 2005.
- [9] J. Craig, *Robótica*, México: Pearson Educación, 2006.
- [10] Ó. Muñoz, "Desarrollo de una consola virtual para el movimiento de robots", Jul. 13. 2018. <https://repositorio.upct.es/xmlui/bitstream/handle/10317/7305/tfg-mu%c3%b1-des.pdf?sequence=1&isAllowed=y> (consultado ag. 8, 2021).
- [11] B. Siciliano, L. Sciavicco, L. Villani y G. Oriolo, *Robotics: modelling, planning and control*, Springer Science & Business Media, 2010.
- [12] P. M. Kebria, S. Al-Wais, H. Abdi y S. Nahavandi, "Kinematic and dynamic modelling of UR5 manipulator," de *2016 IEEE international conference on systems, man, and cybernetics (SMC)*, 2016.
- [13] C. L. Guevara y D. J. Guevara, "Diseño, simulación e implementación de técnicas de control basadas en un modelo de orden reducido y aplicadas al seguimiento de trayectorias para la plataforma robótica Pioneer 3DX", Departamento de Automatización y Control Industrial, Escuela Politécnica Nacional, Quito, Ecuador, Septiembre 2016.
- [14] R. Guamán y W. Lozada, "Control de Posición y Seguimiento de Trayectorias de un Manipulador Móvil de 3 Grados de Libertad", Departamento de Automatización y Control Industrial, Escuela Politécnica Nacional, Quito, Ecuador, Julio 2017.

- [15] J. Luh, M. Walker y R. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Transactions on Automatic Control*, vol. 25, p. 468–474, 1980.
- [16] G. Scaglia, L. Q. Montoya, V. Mut y F. di Sciascio, "Numerical methods based controller design for mobile robots," *Robotica*, vol. 27, p. 269–279, 2009.
- [17] CoppeliaSim, "*CoppeliaSim User Manual*", Coppeliarobotics, <https://www.coppeliarobotics.com/helpFiles/> (consultado ag. 2, 2021).
- [18] CoppeliaSim, "*Remote API modus operandi*", CoppeliaSim Robotics, <https://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm> (consultado nov. 12, 2021).
- [19] B. Wang, C. Yang y Q. Xie, "Human-machine interfaces based on EMG and Kinect applied to teleoperation of a mobile humanoid robot," de *Proceedings of the 10th World Congress on Intelligent Control and Automation*, 2012.
- [20] S. Shin, R. Tafreshi y R. Langari, "EMG and IMU based real-time HCI using dynamic hand gestures for a multiple-DoF robot arm," *Journal of Intelligent & Fuzzy Systems*, vol. 35, p. 861–876, 2018.
- [21] S. Shen, S. Bai, J. Xu y N. Wang, "EMG-Controlled Force Feedback Underwater Manipulator," de *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, 2018.
- [22] L. D. Weiss, J. M. Weiss y J. K. Silver, *Easy EMG e-book: A guide to performing nerve conduction studies and electromyography*, Elsevier Health Sciences, 2015.
- [23] L. J. Myers, M. Lowery, M. O'malley, C. L. Vaughan, C. Heneghan, A. S. C. Gibson, Y. X. R. Harley y R. Sreenivasan, "Rectification and non-linear pre-processing of EMG signals for cortico-muscular analysis," *Journal of neuroscience methods*, vol. 124, p. 157–165, 2003.
- [24] C. Duran, "Optimización y clasificación de señales EMG a través de métodos de reconocimiento de patrones," *ITECKNE*, vol. 10, January 2013.
- [25] İ. Göker, "Detection and Conditioning of EMG," pp. 58-94, January 2014.
- [26] P. Visconti, F. Gaetani, G. Zappatore y P. Primiceri, "Technical Features and Functionalities of Myo Armband: An Overview on Related Literature and Advanced Applications of Myoelectric Armbands Mainly Focused on Arm Prostheses," *International Journal on Smart Sensing and Intelligent Systems*, vol. 11, pp. 1-25, June 2018.
- [27] M. Tomaszewski, "*MyoMex*", <https://github.com/mark-toma/MyoMex> (consultado ag. 9, 2021).
- [28] M. Benálcazar, L. Barona, L. Valdivieso, X. Aguas y J. Zea, "*EMG-EPN-612 Dataset*", <https://doi.org/10.5281/zenodo.4027874> (consultado ag. 10, 2021).
- [29] M. Feldman, "Hilbert Transform, Envelope, Instantaneous Phase, and Frequency," *Encyclopedia of Structural Health Monitoring*, 2009.

- [30] D. Páez Ramírez, J. P. Romero Camacho y J. G. Guarnizo Marin, “UR3 modelo cinemático inverso”.
- [31] R. G. Rivera, R. G. Alvarado, A. Martínez-Rocamora y F. Auat Cheein, “A comprehensive performance evaluation of different mobile manipulators used as displaceable 3D printers of building elements for the construction industry,” *Sustainability*, vol. 12, p. 4378, 2020.
- [32] A. B. Corripio y C. Smith, *Principles and practice of automatic process control*, John Wiley, 1997.
- [33] K. Krishnan y G. Karpagam, “Comparison of PID controller tuning techniques for a FOPDT system,” *International Journal of Current Engineering and Technology*, vol. 4, p. 2667–2670, 2014.
- [34] A. Chico, P. J. Cruz, J. P. Váscquez, M. E. Benalcázar, R. Álvarez, L. Barona y Á. L. Valdivieso, “Hand Gesture Recognition and Tracking Control for a Virtual UR5 Robot Manipulator,” 2021. [En línea]. Available: <https://ieeexplore.ieee.org/document/9590677>.
- [35] A. A. Syed, X.-g. Duan, X. Kong, M. Li, Q. Huang y others, “6-dof maxillofacial surgical robotic manipulator controlled by haptic device,” de *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2012.
- [36] A. M. Naim, K. Wickramasinghe, A. De Silva, M. V. Perera, T. D. Lalitharatne y S. L. Kappel, “Low-cost Active Dry-Contact Surface EMG Sensor for Bionic Arms,” de *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.
- [37] L. Guevara, J. Guevara, O. Camacho, G. Scaglia y A. Rosales, “A new approach of a Numerical Methods Controller for self-regulating processes,” de *2016 IEEE Biennial Congress of Argentina (ARGENCON)*, 2016.
- [38] L. Capito, P. Proaño, O. Camacho, A. Rosales y G. Scaglia, “Experimental comparison of control strategies for trajectory tracking for mobile robots,” *International Journal of Automation and Control*, vol. 10, p. 308–327, 2016.
- [39] R. Campa y H. De La Torre, “Pose control of robot manipulators using different orientation representations: A comparative review,” de *2009 American Control Conference*, 2009.
- [40] CoppeliaSim, “*Robot simulator CoppeliaSim: create, compose, simulate, any robot*”, CoppeliaRobotics, <https://www.coppeliarobotics.com/> (consultado ag. 2, 2021).
- [41] M. Mordhorst, T. Heidlauf y O. Röhrle, “Mathematically modelling surface EMG signals,” *PAMM*, vol. 14, p. 123–124, 2014.
- [42] C. D. Katsis, T. P. Exarchos, C. Papaloukas, Y. Goletsis, D. I. Fotiadis y I. Sarmas, “A two-stage method for MUAP classification based on EMG decomposition,” *Computers in Biology and Medicine*, vol. 37, p. 1232–1240, 2007.

- [43] C. J. De Luca, "Imaging the Behavior of Motor Units by Decomposition of the EMG signal," *Delsys Inc. Boston*, 2008.
- [44] K. Ogata, *Ingeniería de control moderna*, Madrid: Pearson Education, 2010.
- [45] B. Kolman y D. R. Hill, *Álgebra lineal*, Pearson Educación, 2006.
- [46] V. H. Andaluz, F. Roberto, M. Toibero, P. Leica y R. Carelli, "Adaptive coordinated cooperative control of multi-mobile manipulators," *Frontiers in Advanced Control Systems; IntechOpen: London, UK*, p. 163–190, 2012.
- [47] P. Bonato, T. D'Alessio y M. Knaflitz, "A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait," *IEEE Transactions on biomedical engineering*, vol. 45, p. 287–299, 1998.
- [48] M. M. Seron, *Sistemas No Lineales - Notas de Clase*, sep. 2000. <https://www.fceia.unr.edu.ar/control/snl/Apunte.pdf> (consultado sep. 16, 2021).

ANEXOS

ANEXO A. Modelo Cinemático Directo del Robot Manipulador UR5

ANEXO B. Matriz Jacobiana Geométrica del Robot Manipulador UR5

ANEXO C. Manual de Comunicación entre CoppeliaSim y Matlab

ANEXO D. Diagrama de Flujo de la Interfaz Gráfica

ANEXO E. Manual de Usuario de la Interfaz Gráfica

ANEXO A

Modelo Cinemático Directo del Robot Manipulador UR5

El modelo cinemático directo del manipulador robótico UR5 está representado a partir de la siguiente expresión:

$$T_0^6 = \begin{bmatrix} \hat{x}_0^6 & \hat{x}_0^6 & \hat{x}_0^6 & P_0^6 \\ \hat{y}_0^6 & \hat{y}_0^6 & \hat{y}_0^6 & P_0^6 \\ \hat{z}_0^6 & \hat{z}_0^6 & \hat{z}_0^6 & P_0^6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Desarrollando, se tiene:

$$T_0^6 = \begin{bmatrix} \cos(q_6) \sigma_7 - \sin(q_6) \sigma_3 & -\cos(q_6) \sigma_3 - \sin(q_6) \sigma_7 & \sigma_8 & l_6 \sigma_8 + l_4 \sigma_{15} + l_2 \sin(q_1) - l_5 \sigma_3 + l_3 \cos(q_1) \sigma_{19} \\ -\sin(q_6) \sigma_2 - \cos(q_6) \sigma_6 & \sin(q_6) \sigma_6 - \cos(q_6) \sigma_2 & -\sin(q_5) \sigma_{11} - \sigma_1 & l_4 \sigma_{13} - l_6 (\sin(q_5) \sigma_{11} + \sigma_1) - l_2 \cos(q_1) - l_5 \sigma_2 + l_3 \sigma_{19} \sin(q_1) \\ \sin(q_6) \sigma_5 + \cos(q_5) \cos(q_6) \sigma_4 & \cos(q_6) \sigma_5 - \cos(q_5) \sin(q_6) \sigma_4 & \sin(q_5) \sigma_4 & l_1 + l_4 \sigma_9 + l_3 \sigma_{20} + l_5 \sigma_5 + l_6 \sin(q_5) \sigma_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde:

$$\sigma_1 = \cos(q_1) \cos(q_5)$$

$$\sigma_2 = \sigma_{16} \sigma_{13} + \sigma_{17} \sigma_{14}$$

$$\sigma_3 = \sigma_{17} \sigma_{18} + \sigma_{16} \sigma_{15}$$

$$\sigma_4 = \sigma_{17} \sigma_9 + \sigma_{16} \sigma_{10}$$

$$\sigma_5 = \sigma_{17} \sigma_{10} - \sigma_{16} \sigma_9$$

$$\sigma_6 = \cos(q_5) \sigma_{11} - \cos(q_1) \sin(q_5)$$

$$\sigma_7 = \cos(q_5) \sigma_{12} - \sin(q_1) \sin(q_5)$$

$$\sigma_8 = \sin(q_5) \sigma_{12} + \cos(q_5) \sin(q_1)$$

$$\sigma_9 = \cos(q_3) \sigma_{20} + \sigma_{19} \sin(q_3)$$

$$\sigma_{10} = \cos(q_3) \sigma_{19} - \sin(q_3) \sigma_{20}$$

$$\sigma_{11} = \sigma_{16} \sigma_{14} - \sigma_{17} \sigma_{13}$$

$$\sigma_{12} = \sigma_{17} \sigma_{15} - \sigma_{16} \sigma_{18}$$

$$\sigma_{13} = \cos(q_3) \sigma_{19} \sin(q_1) - \sin(q_1) \sin(q_3) \sigma_{20}$$

$$\sigma_{14} = \cos(q_3) \sin(q_1) \sigma_{20} + \sigma_{19} \sin(q_1) \sin(q_3)$$

$$\sigma_{15} = \cos(q_1) \cos(q_3) \sigma_{19} - \cos(q_1) \sin(q_3) \sigma_{20}$$

$$\sigma_{16} = \sin\left(q_4 - \frac{\pi}{2}\right)$$

$$\sigma_{17} = \cos\left(q_4 - \frac{\pi}{2}\right)$$

$$\sigma_{18} = \cos(q_1) \cos(q_3) \sigma_{20} + \cos(q_1) \sigma_{19} \sin(q_3)$$

$$\sigma_{19} = \cos\left(q_2 + \frac{\pi}{2}\right)$$

$$\sigma_{20} = \sin\left(q_2 + \frac{\pi}{2}\right)$$

ANEXO B

Matriz Jacobiana Geométrica del Robot Manipulador UR5

La Matriz Jacobiana geométrica del manipulador robótico UR5 es:

$$J(q) = \begin{bmatrix} J_v(q) \\ J_\omega(q) \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix}$$

Donde la submatriz de velocidad lineal es la siguiente:

$$J_v(q) = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} \end{bmatrix}$$

Donde:

$$\begin{aligned} J_{11} = & l_2 \cos(q_1) + l_6 \cos(q_1) \cos(q_5) + l_3 \sin(q_1) \sin(q_2) + l_4 \cos(q_2) \sin(q_1) \sin(q_3) + l_4 \cos(q_3) \sin(q_1) \sin(q_2) \\ & + l_5 \cos(q_2) \cos(q_3) \sin(q_1) \sin(q_4) + l_5 \cos(q_2) \cos(q_4) \sin(q_1) \sin(q_3) + l_5 \cos(q_3) \cos(q_4) \sin(q_1) \sin(q_2) \\ & - l_5 \sin(q_1) \sin(q_2) \sin(q_3) \sin(q_4) - l_6 \cos(q_2) \cos(q_3) \cos(q_4) \sin(q_1) \sin(q_5) + l_6 \cos(q_2) \sin(q_1) \sin(q_3) \sin(q_4) \sin(q_5) \\ & + l_6 \cos(q_3) \sin(q_1) \sin(q_2) \sin(q_4) \sin(q_5) + l_6 \cos(q_4) \sin(q_1) \sin(q_2) \sin(q_3) \sin(q_5) \end{aligned}$$

$$J_{12} = -\cos(q_1) \left(l_4 \cos(q_2 + q_3) + \frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_3 \cos(q_2) + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{13} = -\cos(q_1) \left(l_4 \cos(q_2 + q_3) + \frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{14} = -\cos(q_1) \left(\frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{15} = l_6 \cos(q_1) \cos(q_2) \cos(q_3) \cos(q_4) \cos(q_5) - l_6 \sin(q_1) \sin(q_5) - l_6 \cos(q_1) \cos(q_2) \cos(q_5) \sin(q_3) \sin(q_4) \\ - l_6 \cos(q_1) \cos(q_3) \cos(q_5) \sin(q_2) \sin(q_4) - l_6 \cos(q_1) \cos(q_4) \cos(q_5) \sin(q_2) \sin(q_3)$$

$$J_{16} = 0$$

$$J_{21} = l_2 \sin(q_1) - l_3 \cos(q_1) \sin(q_2) + l_6 \cos(q_5) \sin(q_1) - l_4 \cos(q_1) \cos(q_2) \sin(q_3) - l_4 \cos(q_1) \cos(q_3) \sin(q_2) \\ - l_5 \cos(q_1) \cos(q_2) \cos(q_3) \sin(q_4) - l_5 \cos(q_1) \cos(q_2) \cos(q_4) \sin(q_3) - l_5 \cos(q_1) \cos(q_3) \cos(q_4) \sin(q_2) \\ + l_5 \cos(q_1) \sin(q_2) \sin(q_3) \sin(q_4) + l_6 \cos(q_1) \cos(q_2) \cos(q_3) \cos(q_4) \sin(q_5) - l_6 \cos(q_1) \cos(q_2) \sin(q_3) \sin(q_4) \sin(q_5) \\ - l_6 \cos(q_1) \cos(q_3) \sin(q_2) \sin(q_4) \sin(q_5) - l_6 \cos(q_1) \cos(q_4) \sin(q_2) \sin(q_3) \sin(q_5)$$

$$J_{22} = -\sin(q_1) \left(l_4 \cos(q_2 + q_3) + \frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_3 \cos(q_2) + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{23} = -\sin(q_1) \left(l_4 \cos(q_2 + q_3) + \frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{24} = -\sin(q_1) \left(\frac{l_6 \cos(q_2 + q_3 + q_4 - q_5)}{2} + l_5 \cos(q_2 + q_3 + q_4) - \frac{l_6 \cos(q_2 + q_3 + q_4 + q_5)}{2} \right)$$

$$J_{25} = l_6 \cos(q_1) \sin(q_5) + l_6 \cos(q_2) \cos(q_3) \cos(q_4) \cos(q_5) \sin(q_1) - l_6 \cos(q_2) \cos(q_5) \sin(q_1) \sin(q_3) \sin(q_4) \\ - l_6 \cos(q_3) \cos(q_5) \sin(q_1) \sin(q_2) \sin(q_4) - l_6 \cos(q_4) \cos(q_5) \sin(q_1) \sin(q_2) \sin(q_3)$$

$$J_{26} = 0$$

$$J_{31} = 0$$

$$J_{32} = \frac{l_6 \sin(q_2 + q_3 + q_4 + q_5)}{2} - \frac{l_6 \sin(q_2 + q_3 + q_4 - q_5)}{2} - l_3 \sin(q_2) - l_5 \sin(q_2 + q_3 + q_4) - l_4 \sin(q_2 + q_3)$$

$$J_{33} = \frac{l_6 \sin(q_2 + q_3 + q_4 + q_5)}{2} - \frac{l_6 \sin(q_2 + q_3 + q_4 - q_5)}{2} - l_5 \sin(q_2 + q_3 + q_4) - l_4 \sin(q_2 + q_3)$$

$$J_{34} = \frac{l_6 \sin(q_2 + q_3 + q_4 + q_5)}{2} - l_5 \sin(q_2 + q_3 + q_4) - \frac{l_6 \sin(q_2 + q_3 + q_4 - q_5)}{2}$$

$$J_{35} = \frac{l_6 (\sin(q_2 + q_3 + q_4 + q_5) + \sin(q_2 + q_3 + q_4 - q_5))}{2}$$

$$J_{36} = 0$$

Finalmente, la submatriz de velocidad angular es la siguiente:

$$J_{\omega}(q) = \begin{bmatrix} J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} \end{bmatrix}$$

Donde:

$$J_{41} = 0$$

$$J_{42} = \sin(q_1)$$

$$J_{43} = \sin(q_1)$$

$$J_{44} = \sin(q_1)$$

$$J_{45} = -\cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_1) \cos(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos(q_1) \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_3) \right) \\ - \sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_1) \cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) - \cos(q_1) \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right)$$

$$J_{46} = \sin(q_5) \left(\cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_1) \cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) - \cos(q_1) \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right) \right. \\ \left. - \sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_1) \cos(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos(q_1) \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_3) \right) \right) + \cos(q_5) \sin(q_1)$$

$$J_{51} = 0$$

$$J_{52} = -\cos(q_1)$$

$$J_{53} = -\cos(q_1)$$

$$J_{54} = -\cos(q_1)$$

$$J_{55} = -\sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_1) - \sin(q_1) \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right) \\ - \cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \sin(q_1) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_1) \sin(q_3) \right)$$

$$J_{56} = -\sin(q_5) \left(\sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \sin(q_1) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_1) \sin(q_3) \right) \right. \\ \left. - \cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_1) - \sin(q_1) \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right) \right) - \cos(q_1) \cos(q_5)$$

$$J_{61} = 1$$

$$J_{62} = 0$$

$$J_{63} = 0$$

$$J_{64} = 0$$

$$J_{65} = \cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) - \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right) - \sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_3) \right)$$

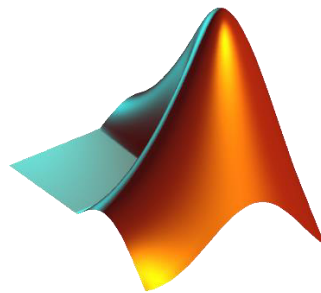
$$J_{66} = \sin(q_5) \left(\cos\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) + \cos\left(q_2 + \frac{\pi}{2}\right) \sin(q_3) \right) + \sin\left(q_4 - \frac{\pi}{2}\right) \left(\cos(q_3) \cos\left(q_2 + \frac{\pi}{2}\right) - \sin(q_3) \sin\left(q_2 + \frac{\pi}{2}\right) \right) \right)$$

ANEXO C

Manual de Comunicación entre CoppeliaSim y MATLAB



CoppeliaSim
from the creators of V-REP



Alex Chico
Septiembre, 2020

A. CONFIGURACIÓN DE MATLAB

Copiar los archivos .m de la siguiente dirección:

C:\ProgramFiles\CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\matlab\matlab

En el siguiente directorio, tal y como se muestra en la Figura C.1:

C:\Program Files\MATLAB\R2019b\bin\win64

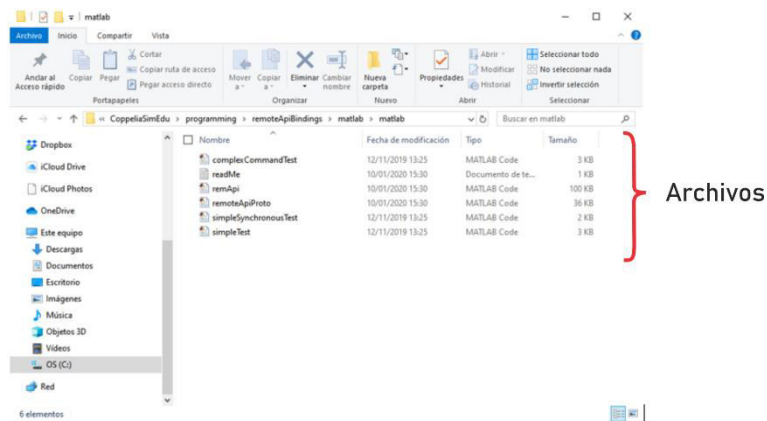


Figura C.1 Archivos de configuración de la API de CoppeliaSim

B. CONFIGURACIÓN DE COPPELIASIM

Siempre que necesitemos crear una escena de CoppeliaSim que interactúe con un programa o script externo a éste, tendremos que configurar la escena tal y como se explica a continuación:

1. Se procede a crear un nuevo script que permita la interconexión entre Matlab y CoppeliaSim. En la ventana scripts se elige **Insert New Script**, ver Figura C.2.

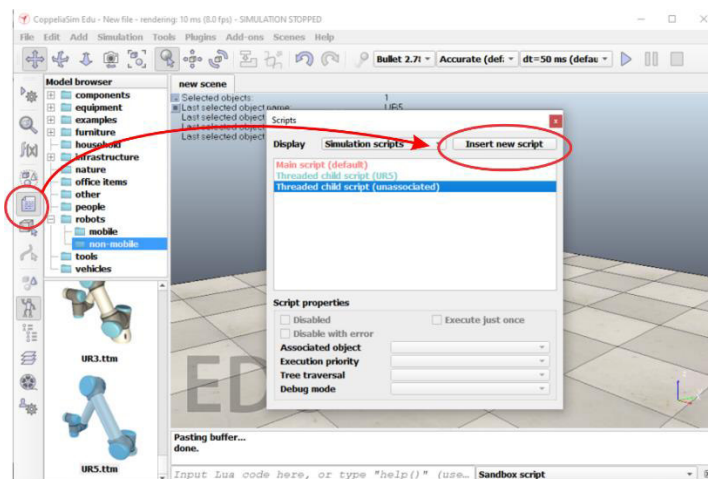
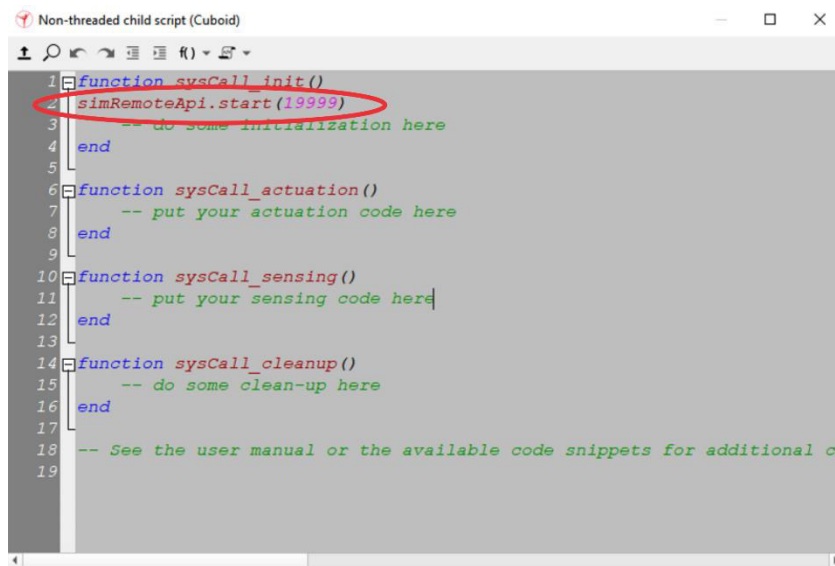


Figura C.2 Insertar nuevo script para la interfaz de comunicación

2. Se abre el nuevo script con doble clic y se añade la instrucción:
`simExtRemoteApiStart(19999);`



```
1 function sysCall_init()
2   simExtRemoteApiStart(19999)
3   -- do some initialization here
4 end
5
6 function sysCall_actuation()
7   -- put your actuation code here
8 end
9
10 function sysCall_sensing()
11   -- put your sensing code here
12 end
13
14 function sysCall_cleanup()
15   -- do some clean-up here
16 end
17
18 -- See the user manual or the available code snippets for additional ca
19
```

Figura C.3 Configuración del puerto de comunicación de la API de CoppeliaSim

C. COMUNICACIÓN ENTRE COPPELIASIM Y MATLAB

1. Configuración del Manipulador Virtual En CoppeliaSim

Una vez creada una escena en CoppeliaSim y hayamos elegido un manipulador virtual es necesario desactivar el script por defecto que lo maneja desde el propio CoppeliaSim. Primero, se selecciona la pestaña scripts y se ubica el child script asociado a nuestro robot. Posteriormente, se selecciona la pestaña deshabilitar, ver Figura C.4.

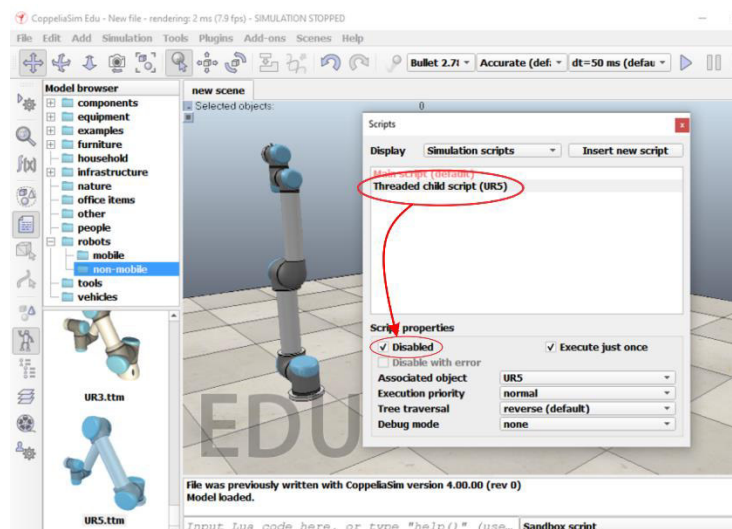


Figura C.4 Desactivación del child script asociado con el modelo del robot UR5

Para acceder a los diferentes elementos que forman nuestra escena en CoppeliaSim y las estructuras de nuestros robots virtuales podemos acceder a la sección Scene Hierarchy. En donde, se desplegarán todos los elementos que forman nuestra escena, incluyendo a los robots y sus partes, como articulaciones del manipulador robótico o motores eléctricos para el caso de un robot móvil. En nuestro manipulador robótico tenemos un motor eléctrico para cada articulación. En el caso del robot UR5, estas se identifican de la siguiente forma:

- UR5_joint1
- UR5_joint2
- UR5_joint3
- UR5_joint4
- UR5_joint5
- UR5_joint6

Entonces, para iniciar con la interfaz de comunicación para controlar estas articulaciones desde Matlab se debe seguir el procedimiento que se muestra a continuación:

2. Conexión

Dentro de nuestro script de Matlab, se deben utilizar las siguientes líneas de código:

```
vrep=remApi('remoteApi');  
vrep.simxFinish(-1);  
clientID=vrep.simxStart('127.0.0.1',19997, true, true, 5000, 5);
```

- Para conectarnos con CoppeliaSim debemos **crear un objeto CoppeliaSim** y por lo tanto debemos **cargar la librería** con todas sus funciones.
- Debemos desconectar la comunicación con **simxFinish** puesto que puede haber problemas para volver a conectarte al mismo puerto.
- Después **inicializamos la simulación** llamando la función **simxStart**.

3. Nodos del Robot

```
[returnCode, art_1] = vrep.simxGetObjectHandle(clientID, 'UR5_joint1',  
vrep.simx_opmode_blocking);
```

- Permite identificar un objeto a través de una etiqueta desde Matlab, por ejemplo: crear una variable que identifique una articulación de un manipulador.

4. Leer posición angular de una articulación

El comando **simxGetJointPosition** permite leer la posición angular de una articulación de CoppeliaSim declarado como un nodo anteriormente. En este caso, se recibe la posición angular de radianes de la articulación 1 del Manipulador robótico UR5 de CoppeliaSim.

```
[returnCode, rotation1] = vrep.simxGetJointPosition(clientID, art_1, vrep.simx_opmode_buffer);
```

5. Fijar velocidad en una articulación

El comando **simxSetJointTargetVelocity** permite fijar una velocidad angular en radianes por segundo a una articulación de CoppeliaSim declarada como un nodo anteriormente. En este caso, se fija una velocidad de 0.5 rad/s en la articulación 1 del Manipulador robótico UR5 de CoppeliaSim.

```
vrep.simxSetJointTargetVelocity(clientID, art_1, 0.5, vrep.simx_opmode_oneshot);
```

6. Desconexión

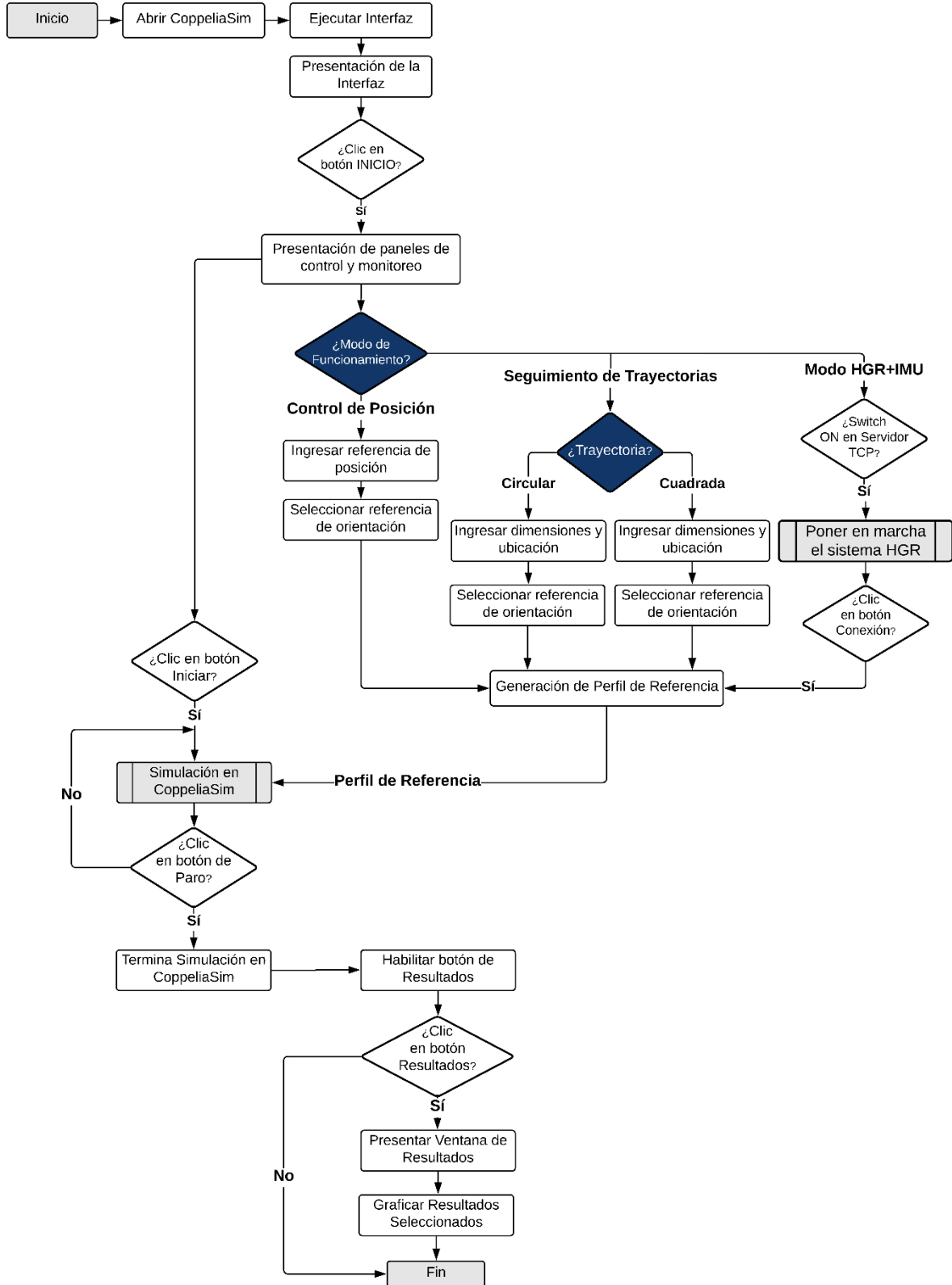
Dentro del script de Matlab, se deben utilizar las siguientes líneas de código para dar por terminada la simulación y conexión entre ambas aplicaciones:

```
vrep.simxAddStatusbarMessage(clientID, 'Comunicación Finalizada', vrep.simx_opmode_blocking);  
vrep.simxStopSimulation(clientID, vrep.simx_opmode_oneshot_wait);  
vrep.simxFinish(clientID);
```

- El comando **simxAddStatusbarMessage** escribe un mensaje en la consola o barra de mensajes de CoppeliaSim.
- El comando **simxStopSimulation** frena la simulación de CoppeliaSim.
- El comando **simxFinish** finaliza el enlace de comunicación entre Matlab y CoppeliaSim

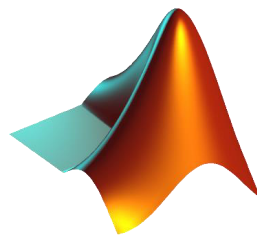
ANEXO D

Diagrama de Flujo de la Interfaz Gráfica



ANEXO E

Manual de Usuario de la Interfaz Gráfica



Alex Chico
Noviembre, 2021

A. ARRANQUE DEL ENTORNO VIRTUAL

1. Instalar en su computador el programa CoppeliaSim Robotics, disponible en el siguiente enlace: <https://www.coppeliarobotics.com/downloads>
2. Abrir CoppeliaSim Robotics, y ejecutar el archivo “UR5 Robot Simulation.ttt” que posee el entorno virtual diseñado para el funcionamiento adecuado del manipulador robótico UR5, que se muestra en la Figura E.1.

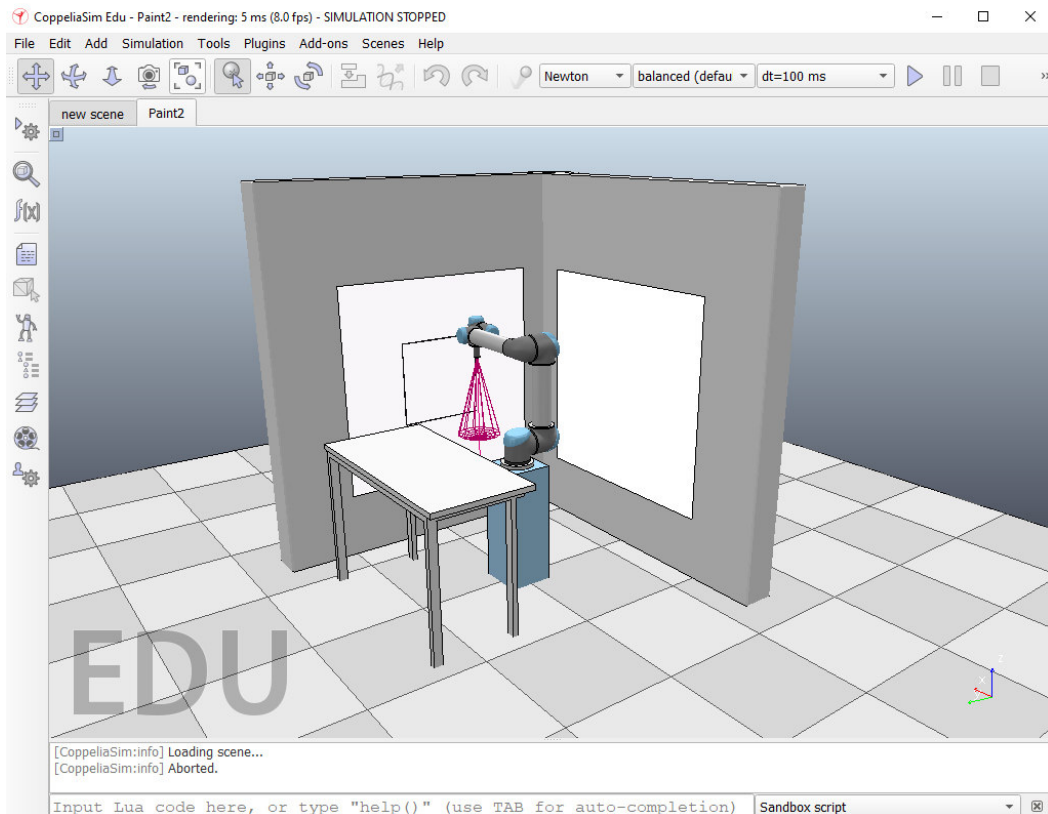


Figura E.1 Entorno virtual desarrollado en CoppeliaSim

B. ARRANQUE DE LA INTERFAZ GRÁFICA

1. Abrir la aplicación App Designer de Matlab y ejecutar el archivo “Aplicacion.mlap”.
2. Se abrirá la interfaz gráfica del proyecto, mostrando en primera instancia su pantalla de presentación, ver Figura E.2. Posteriormente, pulsar el botón **INICIO** para continuar hacia la pantalla principal y empezar a utilizar el sistema.

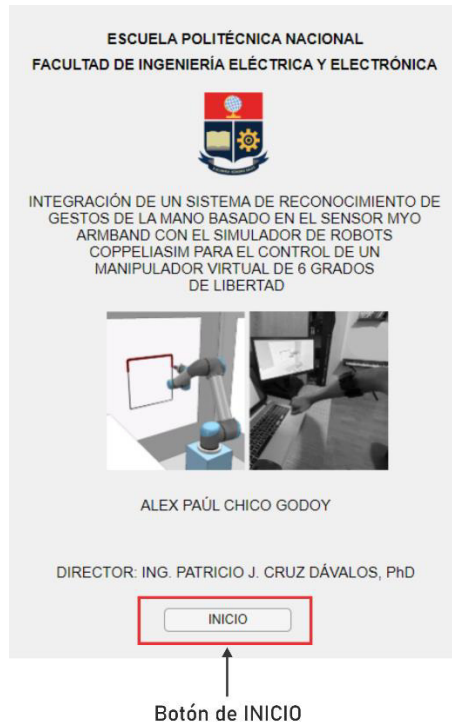


Figura E.2 Pantalla de presentación de la interfaz gráfica del proyecto

3. Se abrirá la pantalla principal de la interfaz, mostrando los paneles de control y monitoreo, que permiten la interacción con el usuario, tal y como se muestra en la Figura E.3.

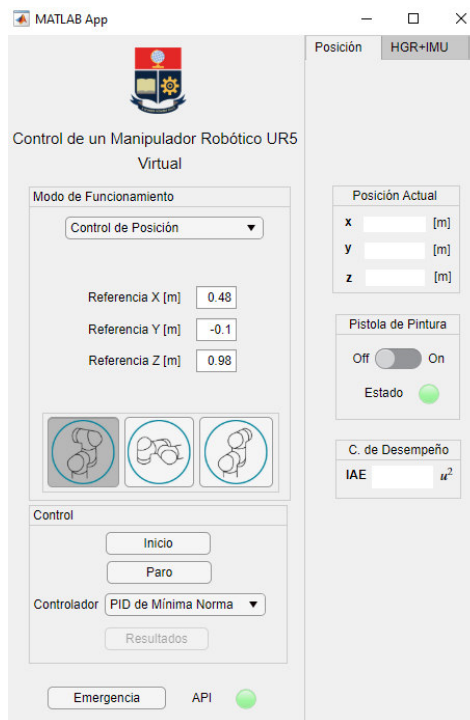


Figura E.3 Panel principal de la interfaz gráfica del proyecto

4. Posteriormente, se debe seleccionar el modo de funcionamiento del robot, que puede ser control de posición, seguimiento de trayectorias, o el modo HGR+IMU, a través de la lista desplegable principal de opciones que se indica en la Figura E.4.

Lista desplegable principal

The screenshot displays the control interface for a virtual UR5 robot. At the top, it features a logo and the text "Control de un Manipulador Robótico UR5 Virtual". The main section is titled "Modo de Funcionamiento" and contains a dropdown menu with "Control de Posición" selected. Below this are three input fields for reference coordinates: "Referencia X [m]" (0), "Referencia Y [m]" (0), and "Referencia Z [m]" (0). There are also three circular icons representing different robot configurations. The "Control" section includes buttons for "Inicio", "Paro", and "Resultados", along with a "Controlador" dropdown set to "PID de Mínima Norma". At the bottom, there is an "Emergencia" button and an "API" status indicator (a green circle). On the right side, there is a "Posición" tab set to "HGR+IMU", a "Posición Actual" section with x, y, and z coordinates, a "Pistola de Pintura" section with an "Off" toggle and a green "Estado" indicator, and a "C. de Desempeño" section with an "IAE" input field.

Figura E.4 Lista desplegable principal de la interfaz gráfica

C. SELECCIÓN DEL CONTROLADOR Y MODO DE FUNCIONAMIENTO

Este sistema permite elegir entre dos técnicas de control para su funcionamiento, un controlador PID de mínima norma o un controlador basado en álgebra lineal (LAC). Estos se eligen en la lista desplegable de controladores, que se indica en la Figura E.5. Sin embargo, la técnica seleccionada por defecto es el controlador PID de mínima norma.



Figura E.5 Lista desplegable de los controladores

Una vez seleccionado el controlador, se debe elegir el modo de funcionamiento del robot y configurar sus parámetros dependiendo de su perfil de referencia.

C.1. CONTROL DE POSICIÓN

1. Una vez seleccionado este modo, ingrese la referencia de posición en las entradas de texto disponibles, ver Figura E.6, que corresponden a la posición deseada del efector final del robot respecto al sistema de referencia del entorno virtual desarrollado. Nota: La unidad de longitud utilizada para representar esta posición, y otros parámetros del sistema, es el metro.

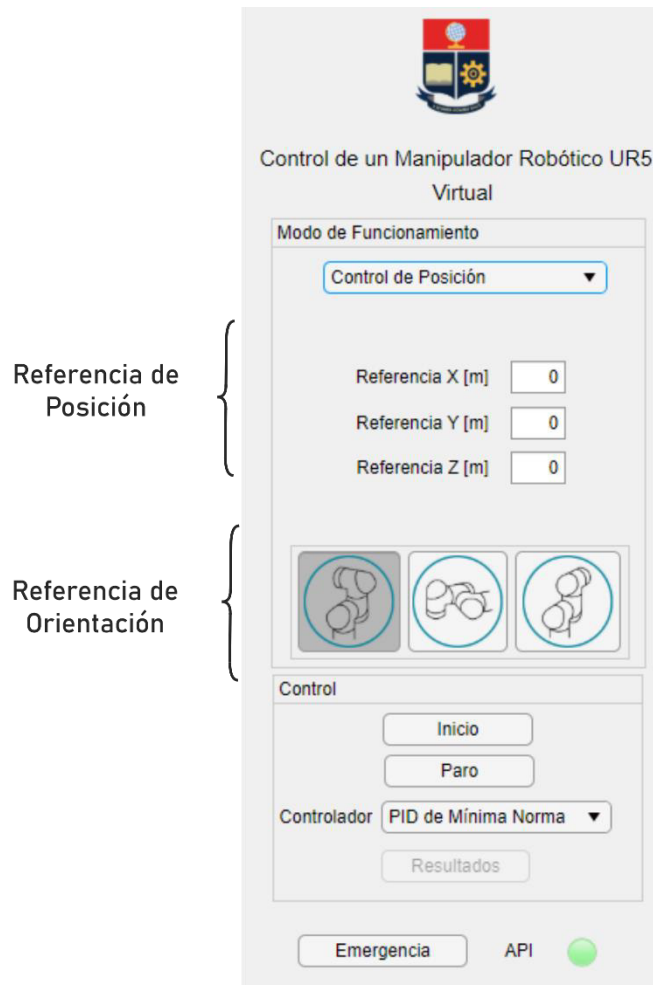


Figura E.6 Referencias de posición y orientación

2. Seleccionar la referencia de orientación de entre las tres diferentes opciones disponibles, ver Figura E.6.

C.2. SEGUIMIENTO DE TRAYECTORIAS

1. Una vez seleccionado el modo de seguimiento de trayectorias, se debe escoger la trayectoria a seguir a través de una lista desplegable que posee dos opciones, trayectoria cuadrada o circular, ver Figura E.7.
2. Una vez seleccionada el tipo de trayectoria a generar, ingrese la ubicación de su centroide a través de las coordenadas X_c , Y_c , Z_c en las entradas de texto disponibles, ver Figura E.7.
3. Ingrese las dimensiones de la trayectoria seleccionada, longitud del radio R para la trayectoria circular, y longitud del lado L para la trayectoria cuadrada.

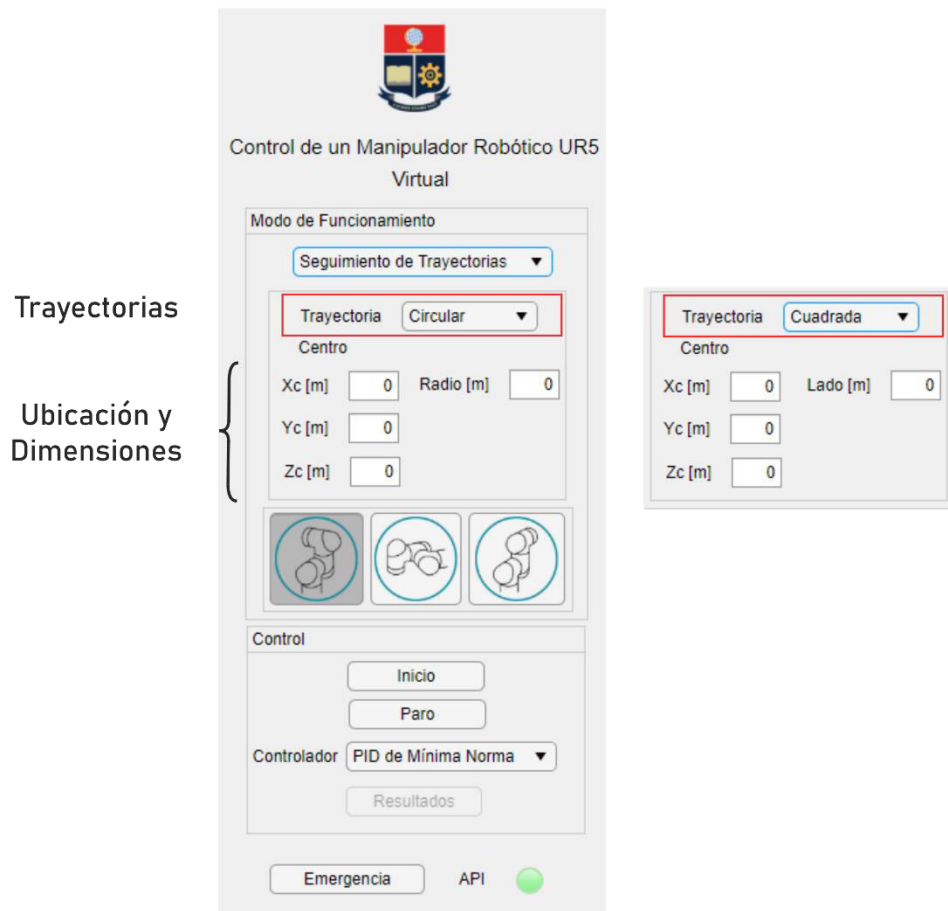


Figura E.7 Modo de seguimiento de trayectorias y configuración

4. Seleccionar la referencia de orientación de entre las tres diferentes opciones disponibles, al igual que en el control de posición.

C.3. MODO HGR+IMU

1. Active el switch del **servidor TCP** para establecer esta instancia de Matlab como cliente al comunicarse con la instancia que ejecuta el sistema de reconocimiento de gestos.
2. Arrancar el sistema de reconocimiento de gestos.
3. Presionar el botón **Conexión** para conectarse con el sensor Myo Armband, ver Figura E.8.

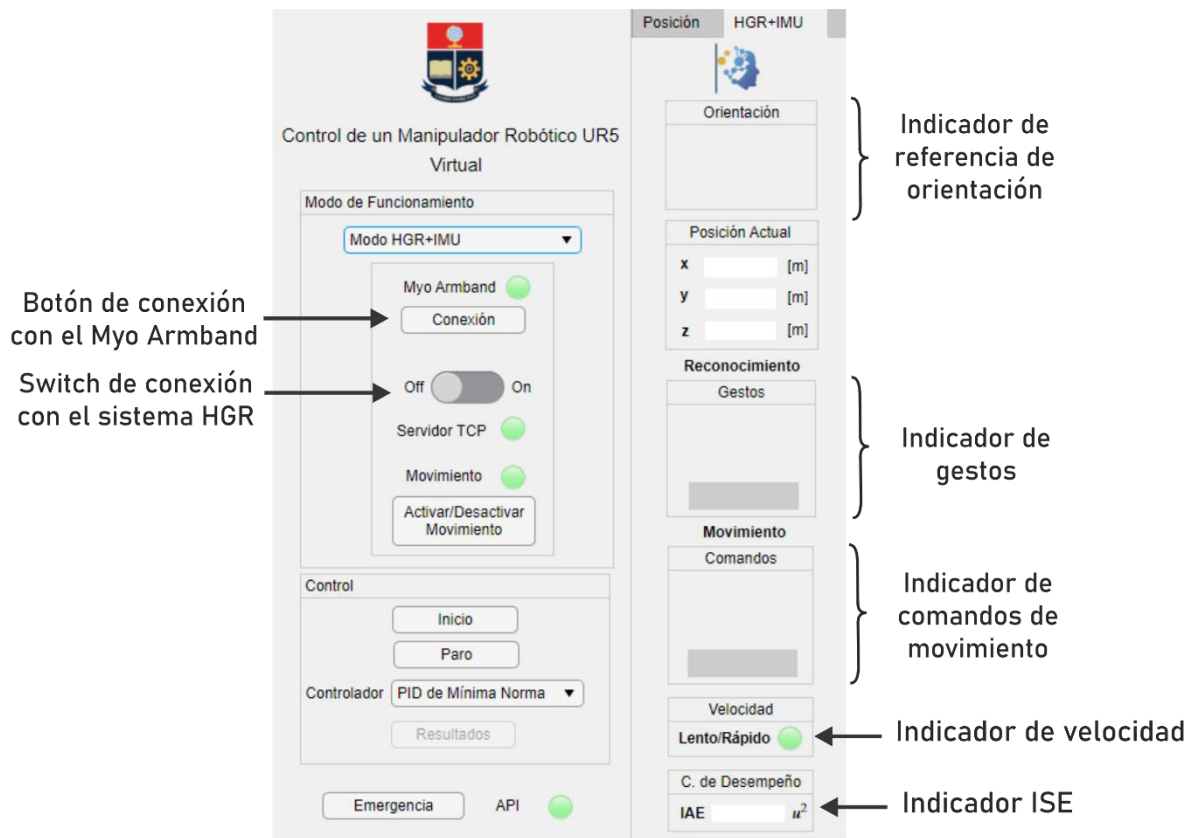


Figura E.8 Modo HGR+IMU y funcionamiento

Nota: Los modos de funcionamiento, control de posición y seguimiento de trayectorias funcionan adecuadamente sin la necesidad de arrancar el sistema de reconocimiento de gestos o el sensor Myo Armband. Sin embargo, para el sistema HGR+IMU, su arranque es obligatorio.

D. ARRANQUE DEL SISTEMA DE RECONOCIMIENTO DE GESTOS

El sensor Myo Armband por defecto cuenta con un cable micro USB y un adaptador Bluetooth, ver Figura E.9. El cable micro USB permite encender y cargar el sensor al conectarse a una computadora, mientras que el adaptador permite comunicar el sensor Myo Armband con nuestro equipo y recibir sus datos de forma inalámbrica.



Figura E.9 Lista desplegable de los controladores

Para arrancar el sistema de reconocimiento de gestos, se deben seguir las siguientes instrucciones:

1. Instalar el Myo Armband Manager, cuyo instalador está disponible en el repositorio de los archivos del sistema.
2. Conectar el adaptador Bluetooth del dispositivo Myo Armband a su computadora.

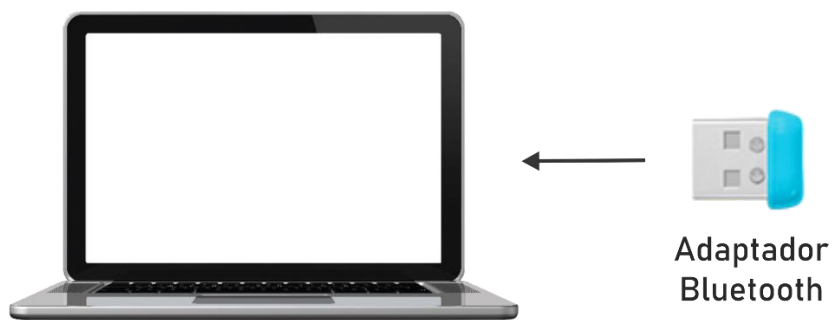


Figura E.10 Conexión del adaptador Bluetooth del Sensor Myo Armband

3. Encender el sensor Myo Armband conectando el sensor a su computadora a través de su cable micro USB, ver Figura E. 11.

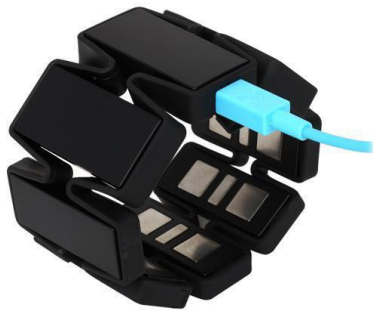


Figura E.11 Conexión del cable micro USB del Sensor Myo Armband

- Una vez que el sensor Myo Armband esté encendido, desconectar su cable micro USB, y colocarse el sensor correctamente en uno de sus antebrazos, ver Figura E.12.

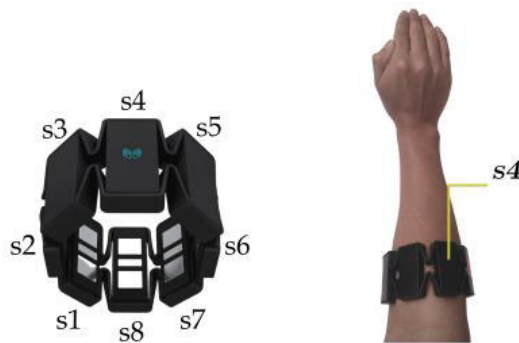
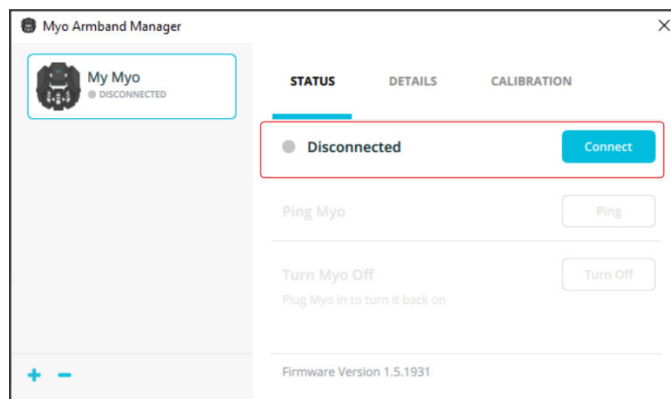


Figura E.12 Posición del Sensor Myo Armband recomendada por el fabricante

- Abrir el Myo Armband Manager, y pulsar el botón **Connect** para establecer correctamente la comunicación entre el sensor Myo Armband y nuestra computadora, ver Figura E.13.



**Botón
Connect**

Figura E.13 Myo Armband Manager

- Abrir una nueva instancia de Matlab, y ejecutar el archivo “Main.m” del sistema de reconocimiento de gestos, tal y como se muestra en la Figura E.14.

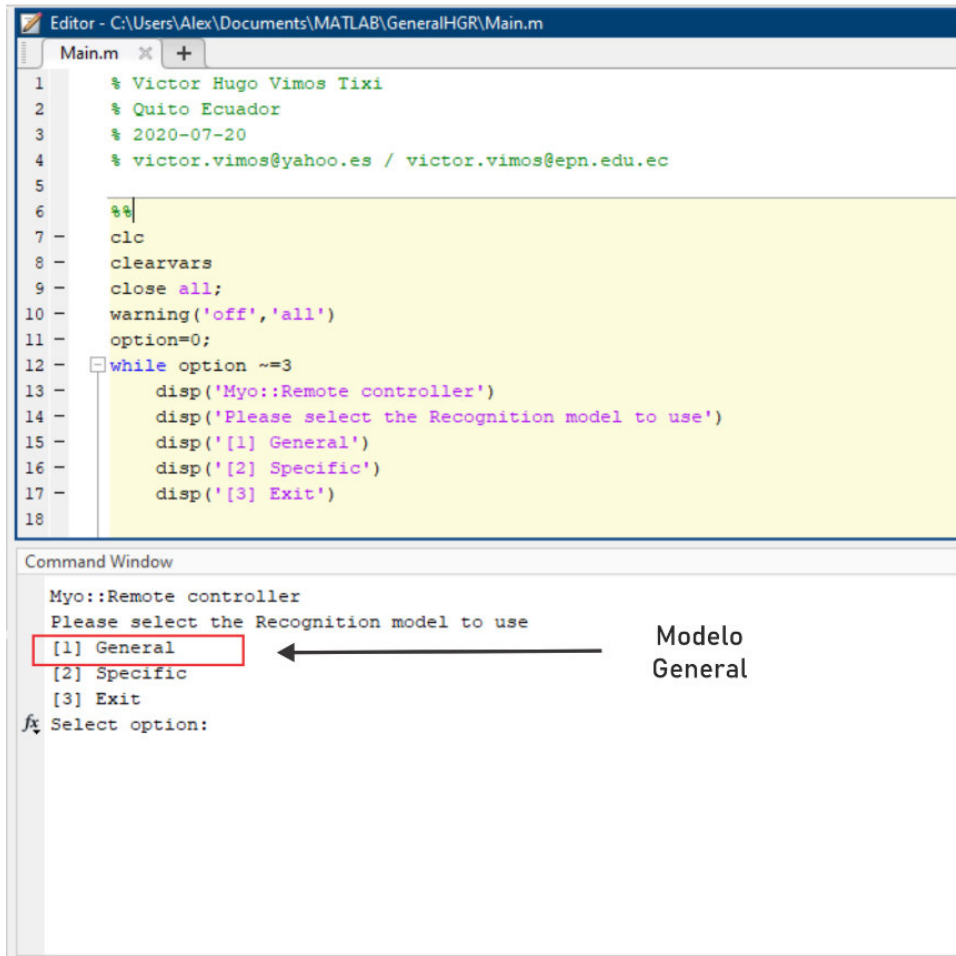


Figura E.14. Menú del sistema de reconocimiento de gestos y su modelo general

7. Seguir las instrucciones mostradas en pantalla, y escoja el modelo general ingresando el número 1 en el Command Window de Matlab y pulsar **ENTER**.
8. Realizar el gesto de sincronización o **WAVEOUT** para entrenar el sistema de reconocimiento y empezar con su funcionamiento.

Nota: La configuración del protocolo de comunicación TCP/IP para el intercambio de información entre esta instancia y la de la interfaz gráfica se realiza automáticamente.

Finalmente, para terminar correctamente el funcionamiento del sistema de reconocimiento de gestos, se debe mantener el gesto **OPEN** durante más de 30 segundos, lo cual desplegará el menú mostrado en la Figura E.14, y posteriormente seleccionar la opción 3 de **Exit**. Adicionalmente, para apagar el sensor Myo Armband, se debe acceder al Myo Armband Manager y presionar el botón **Turn Off**, tal y como se muestra en la Figura E.15.

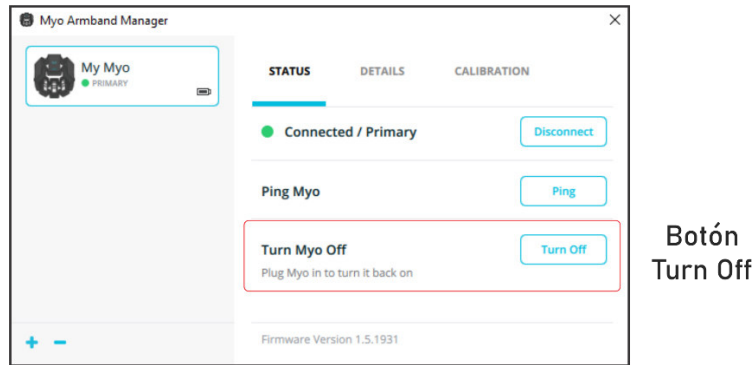


Figura E.15 Myo Armband Manager

E. ARRANQUE DE LA SIMULACIÓN

Para comenzar con la simulación se debe presionar el botón **Inicio** de la interfaz gráfica, arrancando con el sistema de control y la simulación en CoppeliaSim. Adicionalmente, se posee una luz piloto con la etiqueta API que al encenderse indica que la comunicación entre Matlab y CoppeliaSim se estableció correctamente.



Figura E.16 Controles de Inicio y Paro de la simulación

De igual forma, podemos frenar la simulación en todo momento con el botón **Paro**, que se halla en el mismo subpanel que el botón de **Inicio**.

Finalmente, en caso de existir algún problema durante la simulación, se incluye el botón **Emergencia** que frena tanto el lazo de control, como la simulación e interfaz de comunicación entre Matlab y CoppeliaSim, cerrando la interfaz gráfica.

F. PRESENTACIÓN DE RESULTADOS

1. Una vez finalizada la simulación, se activará el botón **Resultados**, indicado en la Figura E.17, que permite abrir una ventana independiente destinada a presentar a detalle los resultados de la última simulación realizada.
2. Presione el botón **Resultados** para obtener las gráficas tanto del seguimiento de las referencias de posición en los ejes x , y y z , como de la norma de los errores de posición y orientación obtenidos.

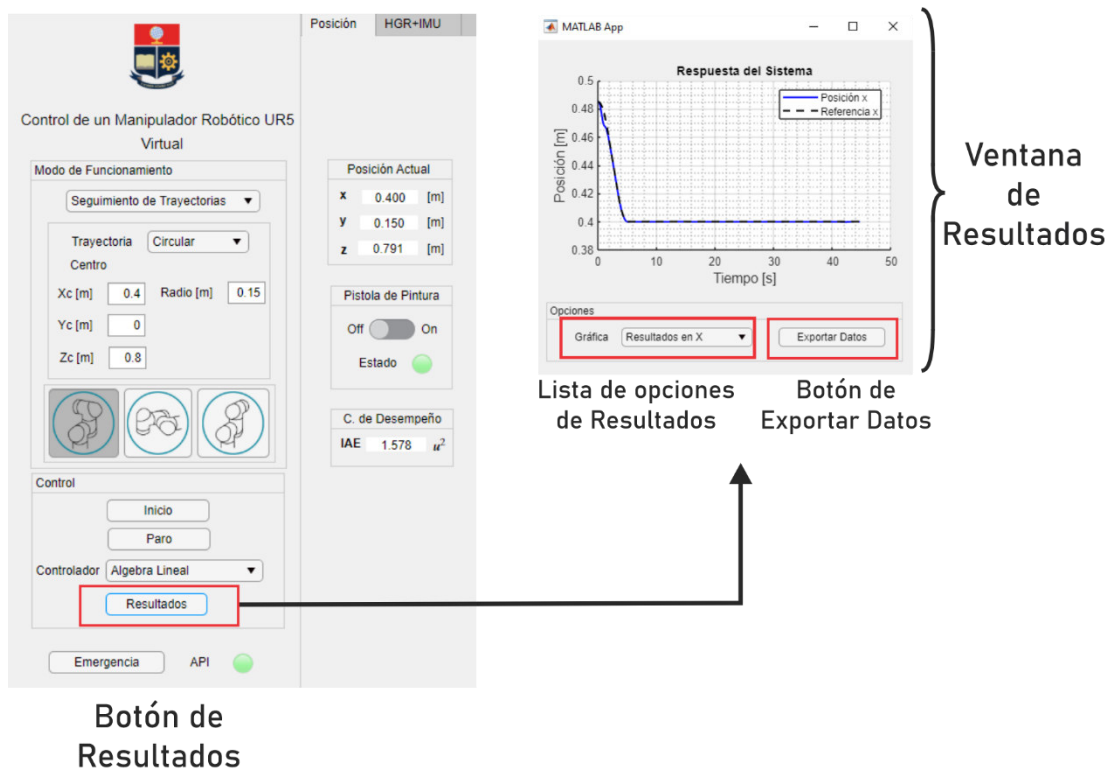


Figura E.17 Ventana de la presentación de resultados de la simulación

3. En la ventana de Resultados, presione el botón **Exportar** para generar un archivo con formato .mat con todos los datos de las referencias de posición, respuestas del sistema y errores calculados resultantes de la simulación anterior.

ORDEN DE EMPASTADO